3-1981

APPLICATION OF INFORMATION THEORY TO THE CONSTRUCTION OF EFFICIENT DECISION TREES

Carlos R.P. Hartmann
Syracuse University, chartman@syr.edu

Pramod Varshney
Syracuse University, varshney@syr.edu

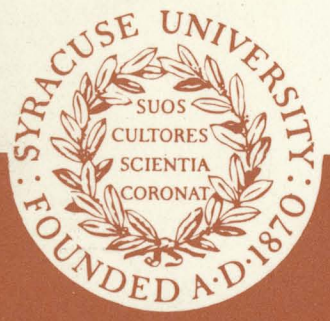
Kishan Mehrotra
Syracuse University, mehrotra@syr.edu

Carl L. Gerberich
Syracuse University

APPLICATION OF INFORMATION THEORY TO THE CONSTRUCTION OF EFFICIENT DECISION TREES*

CARLOS R.P. HARTMANN
PRAMOD K. VARSHNEY
KISHAN G. MEHROTRA
CARL L. GERBERICH

MARCH 81



SCHOOL OF COMPUTER
AND INFORMATION SCIENCE
SYRACUSE UNIVERSITY

APPLICATION OF INFORMATION THEORY TO THE CONSTRUCTION OF EFFICIENT DECISION TREES*

CARLOS R.P. HARTMANN
PRAMOD K. VARSHNEY
KISHAN G. MEHROTRA
CARL L. GERBERICH

C.R.P. Hartmann and K.G. Mehrotra are with the School of Computer and Information Science, Syracuse University, Syracuse, NY 13210

P.K. Varshney is with the Dept. of Electrical and Computer Engineering, Syracuse University, Syracuse, NY 13210

C.L. Gerberich is with IBM Corporation, Poughkeepsie, NY 12601

* This work was partially supported by the National Science Foundation under Grant ENG 79-04826.

TABLE OF CONTENTS

ABSTRACT

This paper treats the problem of conversion of decision tables to decision trees. In most cases, the construction of optimal decision trees is an NP-Complete problem and, therefore, a heuristic approach to this problem is necessary. In our heuristic approach, we apply information theoretic concepts to construct efficient decision trees for decision tables which may include "don't-care" entries. In contrast to most of the existing heuristic algorithms, our algorithm is systematic and has a sound theoretical justification. The algorithm has low design complexity and yet provides us with near-optimal decision trees.

1. INTRODUCTION

In 1948, Shannon [1,2] proposed a mathematical theory to understand the elusive true nature of the communication process and to find its inherent limitations. There, resulted theorems of great power, elegance and generality and these results have blossomed into the field known as information theory. While information theory was primarily developed to deal with the fundamental questions in communication theory it has had a much broader impact. Information theoretic concepts have found widespread applications in such diverse areas as statistics, optimization and population studies. Recently, there is a renewed interest in the application of these concepts to some important problems in the field of computer science as indicated by Toby Berger, in his presentation at the Shannon Theory Workshop on future research directions held at Mount Kisco, New York, during September 12-14, 1979.

In many computer application areas, the design of efficient data processing algorithms is of fundamental importance. One of the problems in this area which has many practical applications is a situation where certain actions or decisions depend on the outcomes of a set of tests. A convenient way of specifying the correspondence between test outcomes and the actions is by means of a decision table. Decision tables have found a widespread application in various areas of data processing, e.g., computer programming, data documentation. In addition, it has applications in the fault-diagnosis of digital systems and artificial intelligence.

In order to formalize a decision process, one prefers a decision table, but when one wants to program it, a decision tree is found to be more suitable. It is, therefore, necessary to devise algorithms for the conversion of decision tables into decision trees. Several algorithms for such conversion have been proposed in the literature [3-30] which are optimal (or near-optimal) according to some criterion. The most frequently considered efficiency measures for such

a conversion are:

1. Storage, i.e., construction of a decision tree with minimum number of nodes (space efficient programs)

2. Time, i.e., construction of a decision tree which minimizes average execution time (execution time efficient programs).

Several algorithms for the construction of optimal decions trees have been proposed. These algorithms are based on two different approaches. Reinwald and Soland [5] have suggested a branch-and-bound approach whereas a dynamic programming approach has been taken by Garey [3], Schumacher and Sevcik [4], Goel [17], and Bayes [23]. These algorithms always guarantee an optimal solution but require an extensive search. For example, in the dynamic programming method of Schumacher and Sevcik, the storage requirement and the execution time grow with the number of binary tests, M, in proportion to 3^M. The branch-and-bound algorithm of Reinwald and Soland is even worse, as pointed out in [4]. This comes to us as no surprise since it has recently been shown that the construction of optimal decision trees in many cases is an NP-complete problem [31,32]. Thus, at present we conjecture that there does not exist an efficient algorithm to find an optimal decision tree (on the supposition that NP $\neq$ P). This result provides us the motivation to find efficient heuristics for constructing near-optimal decision trees.

Most of the heuristic algorithms apply the principle of decomposition. In these algorithms, tests are selected at each stage of the construction of the decision tree according to some criterion. These decomposition algorithms are computationally efficient but, in general, do not generate optimal decision trees. Some of these algorithms use heuristics which are based on information theory concepts (e.g. [10,11,16,18,19,21,28]).

In this paper, we employ an information theoretic approach to the conversion of decision tables to decision trees where decision tables may include "don't care" ("dash") entries. In contrast to most of the existing

heuristic algorithms, our algorithm is systematic and has a sound theoretical justification. The algorithm has low design complexity and yet provides us with near-optimal decision trees. Our approach is to first obtain an upper bound on the efficiency criterion of a given decision tree. Then, a decision tree is designed so as to minimize this upper bound at each step of its construction. In Section 2, we formulate the problem, introduce the notation and present some background material. In Section 3, we obtain bounds on the generalized efficiency measure for a given decision tree. This upper bound is employed in Section 4 for the construction of efficient decision trees. Complexity of this construction is discussed in Section 5 and a summary is presented in Section 6. The concepts are illustrated by means of examples throughout the paper.

2. PRELIMINARIES:

Let $U = \{u_1, \ldots, u_K\}$ be a finite set of unknown objects with an associated probability measure such that $P_U(u_k)$ represents the frequency of occurrence of the object u_k. For each unknown object u_k, there is a corresponding action A_i which must be taken. Let $A = \{A_1, \ldots, A_I\}$ denote the set of all possible actions. Thus, we have an onto mapping $\phi : U \rightarrow A$. Let $\{T_1, \ldots, T_M\}$ be a finite set of tests to be applied to the elements of U. When a test is applied to an object, one of the D possible outcomes can occur, i.e., for a test T_m, $1 \leq m \leq M$, and an object u_k, $1 \leq k \leq K$, we have $T_m(u_k) = d$ where $d \in \{0, \ldots, D-1\}$. We may also assume a cost C_m associated with each test T_m. The problem is to construct an efficient testing algorithm which, for any unknown object of U, uniquely identifies the corresponding action to be taken. In general, ϕ may be a many-to-one mapping. When this is the case, it is not necessary to identify the objects of U in order to be able to identify the action to be taken. It suffices to identify the subsets of U whose elements correspond to the same action. But it may not be achieved due to the constraints imposed by the available set of tests.

A testing algorithm is essentially a D-ary decision tree, and a test is specified at its root node and all other internal nodes. The terminal nodes specify subsets of U whose elements correspond to the same action. It should be observed that two or more of the above subsets at the terminal nodes may

correspond to the same action. The testing algorithm is implemented by first applying the test specified at the root node to the set of unknown objects. If the outcome is (d-1), we take the dth branch from the root node. This procedure is repeated at the root node of each successive subtree until one reaches a terminal node which names an unknown subset of objects whose elements correspond to the same action or the action itself. In this paper, we assume that a testing algorithm always exists. The necessary and sufficient condition for this is given by

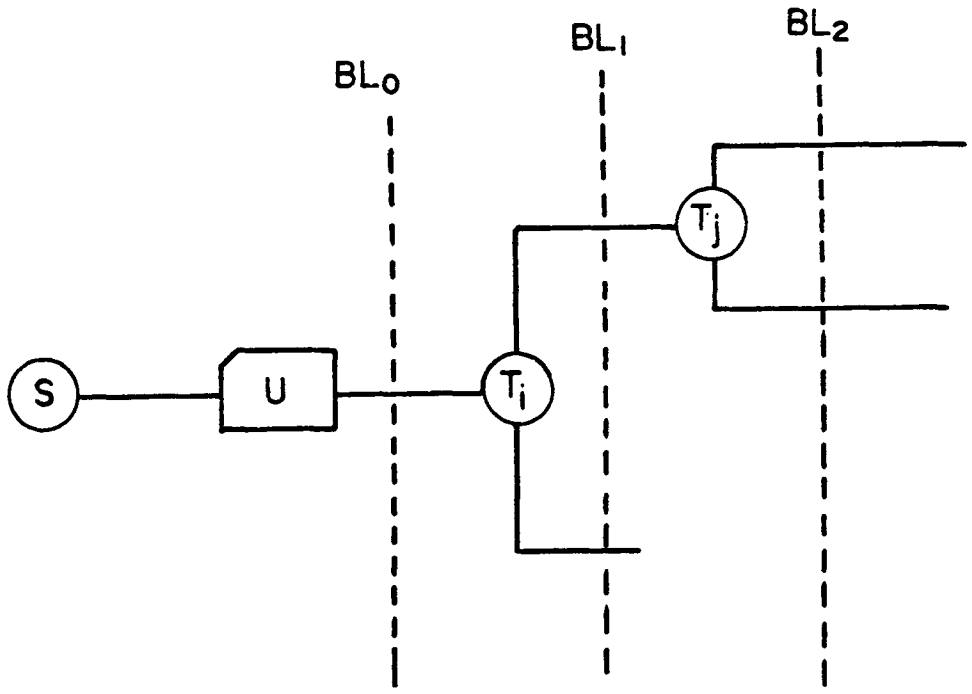
$$(T_1(u_i),\ldots,T_M(u_i)) = (T_1(u_j),\ldots,T_M(u_j))$$

only if

$$\phi(u_i) = \phi(u_j).$$

Testing algorithms, which contain tests that do not distinguish at least two different sets of objects, will not be considered here since these tests may be dropped from the testing algorithm thereby making it more efficient.

In order to define a general efficiency measure, which includes the most frequently used measures such as storage, average cost and average execution time, the notion of branch level at any branch of a decision tree needs to be introduced. Branch level zero (BL_0) is defined prior to the root node of the decision tree. Any branch which has i decision nodes between BL_0 and itself is defined to belong to the branch level i (BL_i). The notion of branch level is illustrated below.



Let s_i, i = 0,1,...,R, be a set of nonnegative integers such that $0 = s_0 < s_1 < \ldots < s_{R-1} < s_R$ where s_R is the length of the longest path in the decision tree, i.e., s_R is equal to the number of decision nodes in the longest path. In the above decision tree, $s_R = 2$. Having introduced the notion of branch level, we are in a position to define our general efficiency measure, $\bar{G}$, which is given by

$$\bar{G} = \sum_{i=1}^{R} g(s_{i-1}, s_i) \tag{1}$$

where

$g(s_{i-1}, s_i)$ is a strictly positive function which can be selected to provide the desired efficiency measure, e.g., storage, average cost and average execution time.

It should be noted that in the special case, when ϕ is a one-to-one mapping, the lower bound on the number of nodes for any D-ary decision tree is given by

$$\text{Number of nodes} \geq 1 + \lceil (K-D)/(D-1) \rceil \tag{2}$$

where $\lceil x \rceil$ represents the smallest integer greater than or equal to x. For a binary decision tree, (2) reduces to an equality and the number of nodes is given by (K-1). Therefore, the storage efficiency measure is not appropriate for the binary case.

Before proceeding further, we illustrate these concepts in the following example.

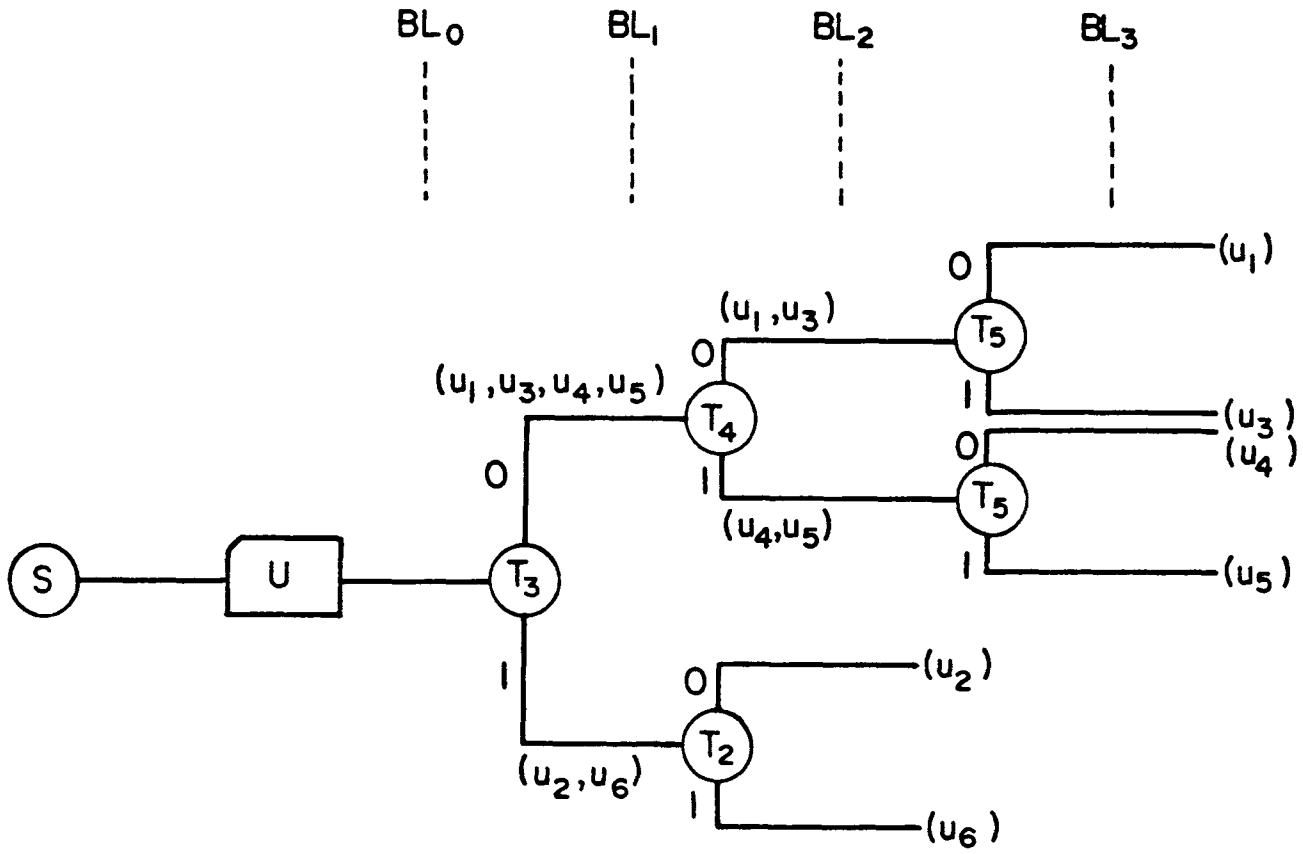
Example 1: Suppose that we have six unknown objects $u_1,\ldots, u_6$. The probabilities of occurrence of these objects are given by

u	u_1	u_2	u_3	u_4	u_5	u_6
$P_U(u)$	0.20	0.05	0.10	0.40	0.20	0.05

We have five tests $T_1,\ldots,T_5$, each having a binary outcome. The following limited-entry decision table gives the result of each test when applied to each of the objects.

T \ u	u_1	u_2	u_3	u_4	u_5	u_6
T_1	0	0	0	1	1	0
T_2	1	0	0	1	1	1
T_3	0	1	0	0	0	1
T_4	0	1	0	1	1	1
T_5	0	1	1	0	1	1

Let us first assume a one-to-one mapping ϕ_1: U → A. In this particular case, identification of actions is the same as the identification of the unknown objects. The following is the flowchart of a testing algorithm for this case.



Next, we wish to interpret our general efficiency measure $\bar{G}$ as the average execution time. If we set $s_i = i$, $i = 1,2,3$, $\bar{G}$ is given by

$$\bar{G} = \sum_{i=1}^{3} g(i-1, i) \tag{3}$$

where

$$g(0,1) = [P_U(u_1) + P_U(u_3) + P_U(u_4) + P_U(u_5)] + [P_U(u_2) + P_U(u_6)]$$

$$g(1,2) = [P_U(u_1) + P_U(u_3)] + [P_U(u_4) + P_U(u_5)] + P_U(u_2) + P_U(u_6)$$

$$g(2,3) = P_U(u_1) + P_U(u_3) + P_U(u_4) + P_U(u_5)$$

For the above choice of s_i's , $g(i-1, i)$ is the sum of probabilities of occurrence of the unknown objects which are associated with the branches of branch level BL_i.

If we have costs associated with the tests, $\bar{G}$ as given by (3) can be interpreted as the average cost. In this case the g's are given by

$$g(0,1) = C_3[P_U(u_1) + P_U(u_3) + P_U(u_4) + P_U(u_5)] + C_3\,[P_U(u_2) + P_U(u_6)]$$

$$g(1,2) = C_4[P_U(u_1) + P_U(u_3)] + C_4\,[P_U(u_4) + P_U\,(u_5)] + C_2\,P_U(u_2) + C_2P_U(u_6)$$

$$g(2,3) = C_5P_U(u_1) + C_5P_U(u_3) + C_5P_U(u_4) + C_5P_U(u_5)\ .$$

For both of the above efficiency measures, consider a different set of integers, $s_1 = 2$ and $s_2 = 3$. In this case, $\overline{G}$ is given by

$$\overline{G} = \sum_{i=1}^{2} g(s_{i-1},\ s_i)$$

where

$g(0,2) = g(0,1) + g(1,2)$ and $g(2,3)$ is the same as before. In general, for the efficiency measures, average cost, average execution time and storage, $g(s_{i-1},\ s_i)$ is given by

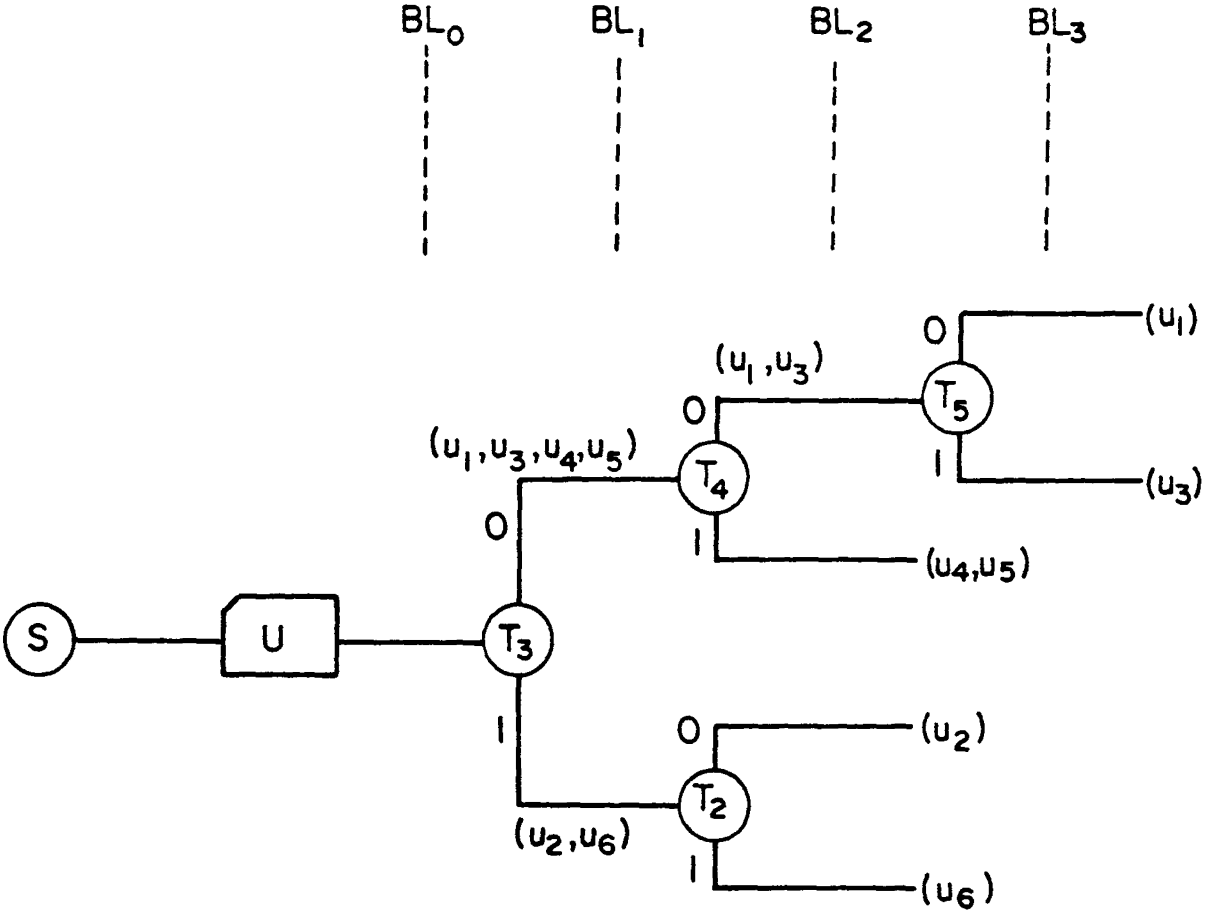
$$g(s_{i-1}\ ,\ s_i) = g(s_{i-1}\ ,\ s_{i-1} + 1) + \ldots + g(s_i - 1\ ,\ s_i).$$

For the mapping ϕ_1 storage is not an appropriate efficiency measure since it is a one-to-one mapping and test outcomes are binary; and, therefore, the number of nodes is five for any testing algorithm.

Let us now assume a many-to-one mapping ϕ_2: U → A which is defined as

u	$\phi_2(u)$
u_1	A_3
u_2	A_3
u_3	A_2
u_4	A_1
u_5	A_1
u_6	A_4

The decision tree obtained for ϕ_1 is a testing algorithm for ϕ_2 also. However, we notice that it is not very efficient since we don't need to distinguish $\{u_1, u_2\}$ and $\{u_4, u_5\}$ as $\phi_2(u_1) = \phi_2(u_2)$ and $\phi_2(u_4) = \phi_2(u_5)$. A more efficient testing algorithm is



In the latter testing algorithm, it turns out that u_1 and u_2 which correspond to the same action are distinguished but u_4 and u_5 are not distinguished. In general, we don't know prior to the construction of the decision tree which unknown objects corresponding to the same action will be distinguished and which will not.

In the case of a many-to-one mapping, storage is an appropriate efficiency measure because the number of nodes is not the same for all testing

algorithms as illustrated by the above decision trees. In order to interpret $\overline{G}$ as the storage efficiency measure, we may set $s_i = i$, i=1,2,3, and $\overline{G}$ is again given by (3) where g(i-1, i) is equal to the number of decision nodes between BL_{i-1} and BL_i. The average execution time and average cost can be obtained as before.

In the literature (e.g. [6], [26]), "don't care" or "dash" has been used to reduce the size of decision tables. A dash (-) may appear as an entry in the decision table. If $T_m(u_k)$ = - in the decision table, then $T_m(u_k)$ can be any one of the D possible outcomes as illustrated in Example 2.

Example 2: Let us consider the random variable U, the limited-entry decision table and the mapping ϕ_2 of Example 1. Note that $\phi_2(u_4) = \phi_2(u_5)$ and $T_m(u_4) = T_m(u_5)$, m=1,...,4, but $T_5(u_4) \neq T_5(u_5)$. We may, therefore, reduce the size of the decision table by combining u_4 and u_5 into u_{4+5} with $P_U(u_{4+5}) = P_U(u_4) + P_U(u_5) = 0.60$ and $T_1(u_{4+5}) = 1$, $T_2(u_{4+5}) = 1$, $T_3(u_{4+5}) = 0$, $T_4(u_{4+5}) = 1$ and $T_5(u_{4+5}) = -$. In this case, even though $\phi_2(u_1) = \phi_2(u_2)$, we cannot combine u_1 and u_2 into a single unknown object because more than one test outcomes differ from each other. The random variable U, the limited-entry decision table and the mapping ϕ_2 for the reduced problem are given by

u	u_1	u_2	u_3	u_{4+5}	u_6
$P_U(u)$	0.20	0.05	0.10	0.60	0.05

T \ u	u_1	u_2	u_3	u_{4+5}	u_6
T_1	0	0	0	1	0
T_2	1	0	0	1	1
T_3	0	1	0	0	1
T_4	0	1	0	1	1
T_5	0	1	1	-	1

u	$\phi_2(u)$
u_1	A_3
u_2	A_3
u_3	A_2
u_{4+5}	A_1
u_6	A_4

It should be pointed out that the introduction of dashes to reduce the size results in an information loss. For instance, in Example 2, if T_5 is applied to u_{4+5}, the probability that the test outcome is zero is unknown. Therefore, whenever possible we should keep all the information so that we may be able to construct a more efficient testing algorithm. However, in many situations the given decision table may already contain dashes and the lost information cannot be recovered. In this paper, we shall address both situations; when all the information is available and when it is not.

Now we define the concept of entropy which will be used in this paper. Let us consider a discrete random variable X taking on values $\{x_1, x_2, \ldots, x_I\}$. Let $P_X(x_i)$ denote the probability of the event $\{X = x_i\}$. The average uncer-

tainity (entropy) of X is defined as

$$H(X) = -\sum_{i=1}^{I} P_X(x_i)\log P_X(x_i). \tag{4}$$

In the next section, we derive bounds on $\overline{G}$.

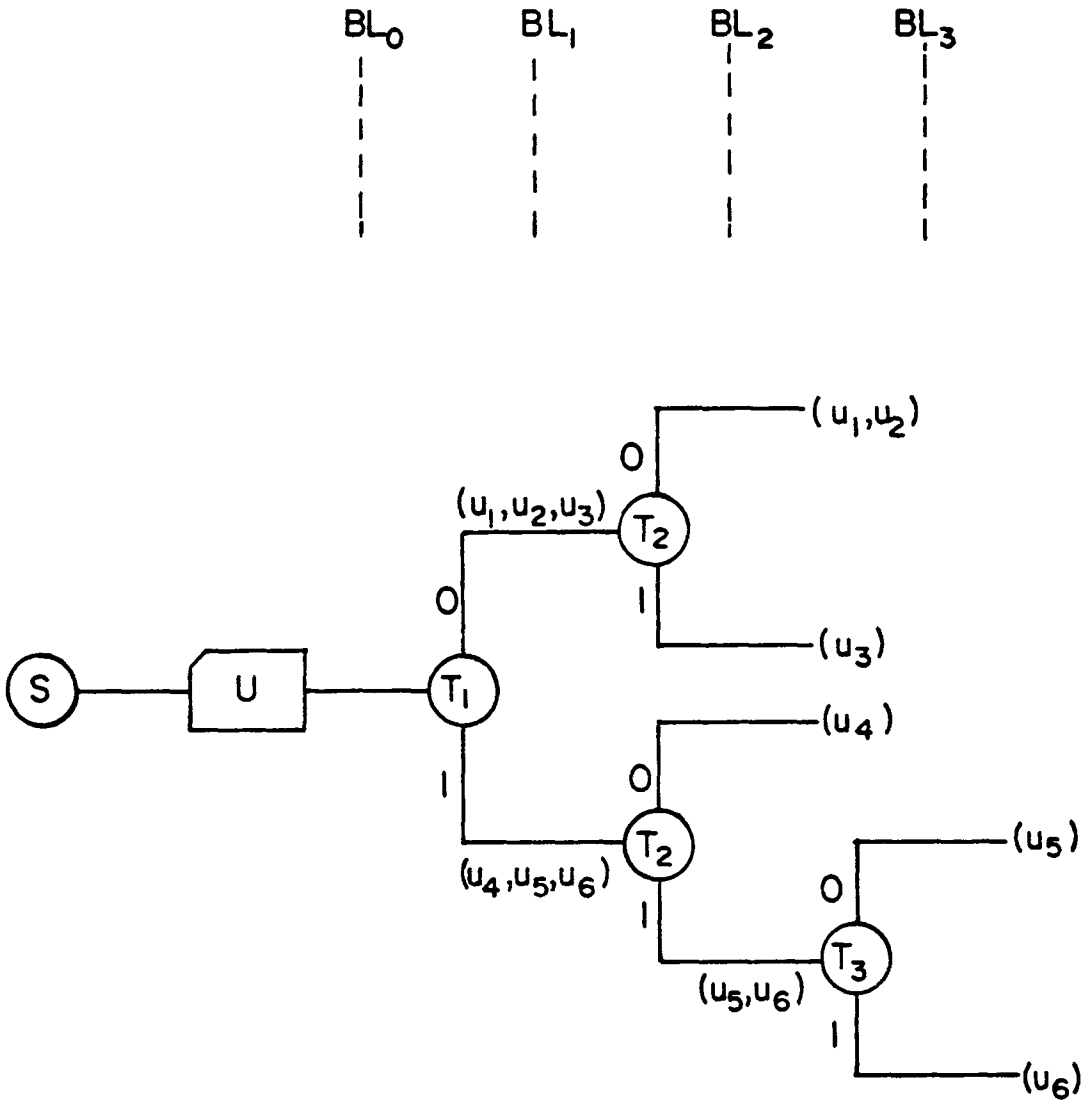
3. BOUNDS ON $\bar{G}$

In this section, we derive bounds on the generalized efficiency measure, $\bar{G}$, for a given decision tree. For notational convenience, we shall only consider the binary case (D=2). However, the results can be extended easily for any value of D. The upper bound will be used in the next section for the construction of efficient decision trees. For the sake of clarity, we first obtain bounds on $\bar{G}$ for a specific example. Bounds on $\bar{G}$ for the general case are derived later.

Example 3: Let us consider the set of six unknown objects $u_1,\ldots, u_6$ and the associated mapping ϕ which is given by

u	$\phi(u)$
u_1	A_1
u_2	A_1
u_3	A_2
u_4	A_3
u_5	A_4
u_6	A_1

Let $P_U(u_k)$ denote the probability of occurrence of u_k as before. Consider the following decision tree which identifies the actions to be taken.



In this example, let us assume $s_i = i$, $i=0,\ldots, 3$. Let $H(BL_i)$ denote the entropy of the branch level BL_i. In this case, we have

$$H(BL_0) = E(u_{126}, u_3, u_4, u_5)$$

$$H(BL_1) = P_U(u_{123})\ E(u_{12}, u_3) + P_U(u_{456})\ E(u_4, u_5, u_6)$$

$$H(BL_2) = P_U(u_{12})\ E(u_{12}) + P_U(u_3)\ E(u_3) + P_U(u_4)\ E(u_4) + P_U(u_{56})\ E(u_5, u_6)$$

and

$$H(BL_3) = P_U(u_5)\ E(u_5) + P_U(u_6)\ E(u_6)$$

where the following notations are used.

$$P_U(u_{ijk}) = P_U(u_i) + P_U(u_j) + P_U(u_k)$$

and (5)

$$E(u_{ijk}, u_\ell, u_{mn}) = - \frac{P_U(u_{ijk})}{P_U(u_{ijk\ell mn})} \log \frac{P_U(u_{ijk})}{P_U(u_{ijk\ell mn})}$$

$$- \frac{P_U(u_\ell)}{P_U(u_{ijk\ell mn})} \log \frac{P_U(u_\ell)}{P_U(u_{ijk\ell mn})}$$

$$- \frac{P_U(u_{mn})}{P_U(u_{ijk\ell mn})} \log \frac{P_U(u_{mn})}{P_U(u_{ijk\ell mn})}$$

We note that $H(BL_i)$ is nothing but a conditional entropy [30].

It is important to note that the objects which correspond to the same action are treated as single objects at various branch levels. For example, in $H(BL_0)$, u_1, u_2 and u_6 are considered to be a single object whereas in $H(BL_1)$ only u_1 and u_2 are considered together because test T_1 is such that it distinguishes u_6 from u_1 and u_2. By definition $E(u_k) = 0$ and, therefore, $H(BL_3) = 0$. We can thus write $H(BL_0) = H(BL_0) - H(BL_1) + H(BL_1) - H(BL_2) + H(BL_2) - H(BL_3)$ or,

$$H(BL_0) = \left(\frac{H(BL_0) - H(BL_1)}{g(0,1)} \right) g(0,1) + \left(\frac{H(BL_1) - H(BL_2)}{g(1,2)} \right) g(1,2)$$

$$+ \left(\frac{H(BL_2) - H(BL_3)}{g(2,3)} \right) g(2,3) \quad . \tag{6}$$

Let

$$H_{min}(s_1,s_2,s_3) = \min_i \left(\frac{H(BL_{i-1}) - H(BL_i)}{g(i-1, i)} \right)$$

and

$$H_{max}(s_1,s_2,s_3) = \max_i \left(\frac{H(BL_{i-1}) - H(BL_i)}{g(i-1, i)} \right)$$

It follows from the results in the Appendix A that $((H(BL_{i-1}) - H(BL_i))/g(i-1, i)) > 0$ for i=1,2,3, and, therefore, we have

$$H_{min}(s_1,s_2,s_3)\ \bar{G} \leq H(BL_0) \leq H_{max}(s_1,s_2,s_3)\ \bar{G}$$

where

$$\bar{G} = \sum_{i=1}^{3} g(i-1, i).$$

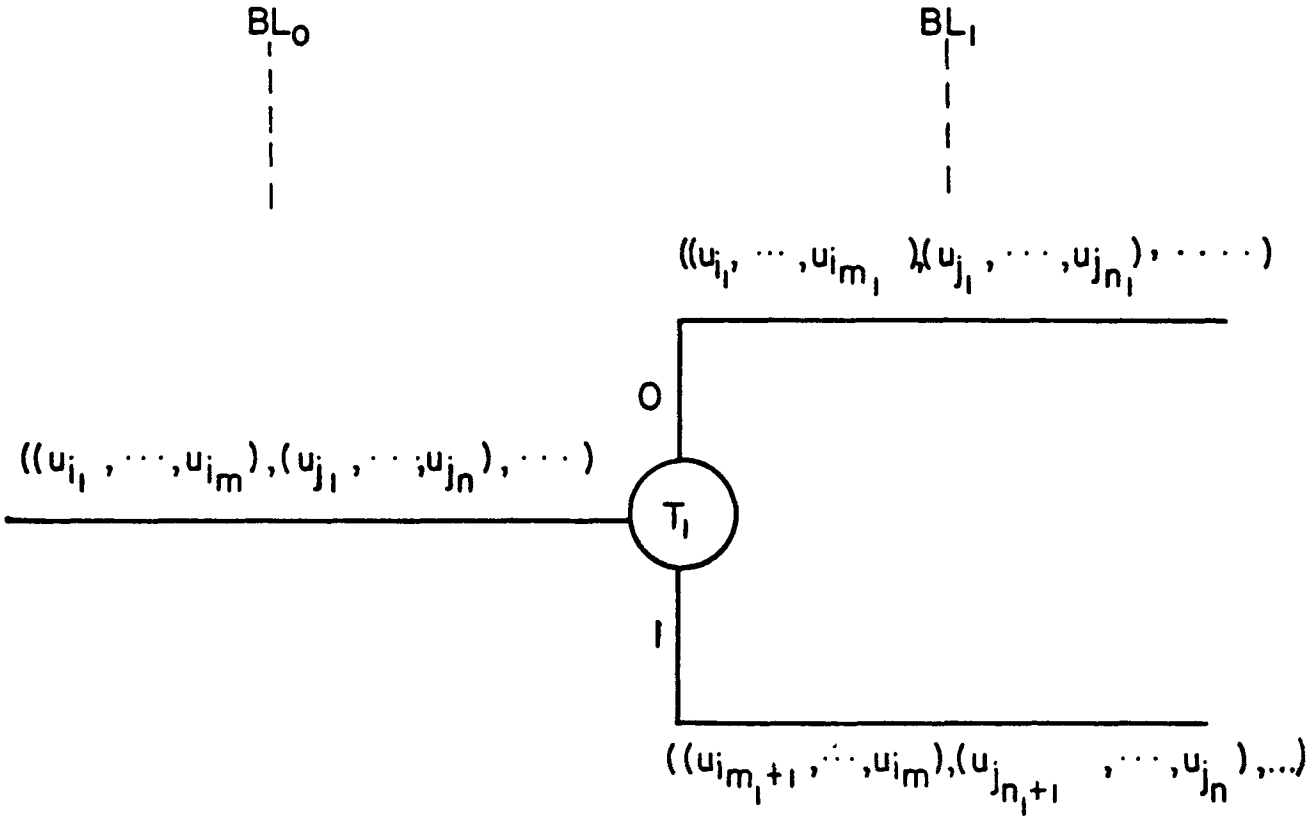
Consequently, we have the following bounds on $\bar{G}$.

$$\frac{H(BL_0)}{H_{max}(s_1,s_2,s_3)} \leq \bar{G} \leq \frac{H(BL_0)}{H_{min}(s_1,s_2,s_3)}$$

Next, we outline the procedure to obtain the bounds for the general case. Let $u_1,\ldots, u_K$ and ϕ denote the set of unknown objects and the associated mapping respectively where $\phi(u_{i_1}) = \phi(u_{i_2}) = \ldots = \phi(u_{i_m})$, $\phi(u_{j_1}) = \phi(u_{j_2}) = \ldots = \phi(u_{j_n})$, etc. Consider a given binary decision tree which identifies the actions to be taken. Let s_i, i=0,1,...,R, be a set of nonnegative integers such that $0 = s_0 < s_1 < \ldots < s_{R-1} < s_R$ where s_R is the length of the longest path in the decision tree under consideration. As in Example 3 using the notations of (5), we may write

$$H(BL_0) = E(u_{i_1\ i_2\ldots i_m}, u_{j_1\ j_2\ldots j_n}, \ldots)$$

Assume that the first test partitions the sets of objects as shown below.



Then,

$$H(BL_1) = P_U(u_{i_1 \cdots i_{m_1} j_1 \cdots j_{n_1} \cdots}) \; E(u_{i_1 \cdots i_{m_1}}, u_{j_1 \cdots j_{n_1}}, \ldots)$$

$$+ P_U(u_{i_{m_1+1} \cdots i_m j_{n_1+1} \cdots j_n \cdots}) \; E(u_{i_{m_1+1} \cdots i_m}, u_{j_{n_1+1} \cdots j_n}, \ldots)$$

All of the $H(BL_i)$ can be obtained in an analogous fashion. Once again we should emphasize that the objects which correspond to the same action are treated as single objects at various branch levels. Since $H(BL_{s_R}) = 0$, we may write

$$H(BL_0) = F(0,s_1)\, g(0,s_1) + F(s_1,s_2)\, g(s_1,s_2) + \ldots + F(s_{R-1}, s_R)\, g(s_{R-1}, s_R) \tag{7}$$

where we use the notation

$$F(s_{i-1}, s_i) = \frac{H(BL_{s_{i-1}}) - H(BL_{s_i})}{g(s_{i-1}, s_i)}$$

As shown in the Appendix A, $H(BL_{s_{i-1}}) - H(BL_{s_i}) \geq 0$, which implies that $F(s_{i-1}, s_i) \geq 0$. Thus, we have the following bounds on $\bar{G}$:

$$\frac{H(BL_0)}{H_{max}(s_1,\ldots,s_R)} \leq \bar{G} \leq \frac{H(BL_0)}{H_{min}(s_1,\ldots,s_R)} \tag{8}$$

where

$$H_{min}(s_1,\ldots,s_R) = \min_i F(s_{i-1}, s_i)$$

and

$$H_{max}(s_1,\ldots,s_R) = \max_i F(s_{i-1}, s_i)$$

Next, we consider a property of this upper bound which will be found useful in the selection of s_i's during the construction of efficient decision trees. Once again we consider a given decision tree which identifies the actions to be taken. Let s_i, i=0,1,...,R, and s'_j, j=0,1,...,Q, be two sets of non-negative integers with $Q \geq R$ such that $0 = s_0 < s_1 < \ldots < s_{R-1} < s_R$ and $0 = s'_0 < s'_1 < \ldots < s'_{Q-1} < s'_Q$, where s_R is equal to s'_Q, and it is the length of the longest path in the decision tree under consideration. Let $S_R = \{s_0, s_1,\ldots, s_R\}$ and $S_Q = \{s'_0, s'_1,\ldots,s'_Q\}$. We assume that $S_R \cap S_Q = S_R$. This property of the upper bound is proved for those functions g which satisfy

$$g(\alpha, \gamma) \leq g(\alpha,\beta) + g(\beta,\gamma) \tag{9}$$

where α, β and γ are nonnegative integers such that $0 \leq \alpha < \beta < \gamma \leq s_R$. We note that the most commonly used efficiency measures average cost, average execution time and storage satisfy (9). Now we are in a position to prove the following theorem.

Theorem 1: Given any decision tree, a function g satisfying (9) and the sets S_R and S_Q satisfying the above properties, we have

$$H_{min}(s_1', \ldots, s_Q') \leq H_{min}(s_1, \ldots, s_R).$$

Proof: Let $H_{min}(s_1,\ldots,s_R)$ be attained at $i=\theta$, i.e., $H_{min}(s_1,\ldots,s_R) = F(s_{\theta-1}, s_\theta)$.

We prove the theorem by contradiction. Let us assume that

$$H_{min}(s_1', \ldots, s_Q') > H_{min}(s_1, \ldots, s_R)$$

As shown in Appendix A, $H_{min}(s_1, \ldots, s_R) \geq 0$ and, therefore,

$$H_{min}(s_1', \ldots, s_Q') > 0. \tag{10}$$

Since $S_R \cap S_Q = S_R$, we may write $s_{\theta-1} = s'_{\alpha_1}$ and $s_\theta = s'_{\alpha_m}$ where s'_{α_1} and s'_{α_m} are in S_Q.

Consider the elements $s'_{\alpha_2}, \ldots, s'_{\alpha_{m-1}}$ of S_Q which are greater than s'_{α_1} but less than s'_{α_m}. We may now express

$$H(BL_{s_{\theta-1}}) - H(BL_{s_\theta}) = \sum_{\ell=1}^{m-1} F(s'_{\alpha_\ell}, s'_{\alpha_{\ell+1}})\, g(s'_{\alpha_\ell}, s'_{\alpha_{\ell+1}}) \tag{11}$$

Let

$$F_m = \min_{1\leq\ell\leq m-1} F(s'_{\alpha_\ell}, s'_{\alpha_{\ell+1}}),$$

then,

$$H(BL_{s_{\theta-1}}) - H(BL_{s_\theta}) \geq F_m \sum_{\ell=1}^{m-1} g(s'_{\alpha_\ell}, s'_{\alpha_{\ell+1}})$$

Using (9) and (10), we may write

$$H(BL_{s_{\theta-1}}) - H(BL_{s_\theta}) \geq F_m\, g(s_{\theta-1}, s_\theta).$$

From the above, we conclude that

$$H_{min}(s_1', \ldots, s_Q') > H_{min}(s_1, \ldots, s_R) \geq F_m$$

which is a contradiction since

$$F_m \geq H_{min}(s_1', \ldots, s_Q')$$

and thus we have the desired result.

Q.E.D.

For the special case, $s_i' = i$, we have the following corollary.

Corollary 1:

$$H_{min}(1,2,\ldots,Q) \leq H_{min}(s_1,\ldots,s_R).$$

In addition, we can obtain a general lower bound on $\overline{G}$ for the efficiency measures average execution time, average cost and storage. This lower bound depends only upon the probability assignment and the mapping ϕ. In order to obtain the lower bound for the first two efficiency measures, we define a new random variable V which takes on values A_i, i=1,..., I, where A_i are the actions as defined earlier. The associated probability measure is $P_V(A_i)$ which is the sum of all $P_U(u_j)$ such that $\phi(u_j) = A_i$. Construct the Huffman code for the random variable V and denote its average length by $\overline{W}_{Huff}$. Average execution time is proportional to the average path length, $\overline{W}$, of the decision tree and, therefore,

$$\overline{W} \geq \overline{W}_{Huff}$$

as seen in [26]. Let $C_{min} = \min_m C_m$, then the average cost, $\overline{C}$, satisfies

$$\overline{C} \geq C_{min}\ \overline{W}_{Huff}.$$

For the storage criterion, from (2),

$$\text{Number of nodes} \geq 1 + \left\lceil (I-D)/(D-1) \right\rceil$$

where I is the number of distinct actions.

The upper bound derived above is employed in the next section for the construction of efficient decision trees.

4. CONSTRUCTION OF EFFICIENT DECISION TREES

4.1 Tables Without Dashes

The basic objective during the construction of decision trees is to minimize $\bar{G}$. As mentioned earlier, in many cases it is impractical to construct an optimum decision tree because it is an NP-complete problem. Therefore, it is desirable to find heuristic yet systematic procedures for an efficient construction. The systematic approach that we follow here is to minimize the upper bound, obtained in the previous section, at each step during the construction of decision trees. For a given decision tree, this upper bound has been shown to be

$$\bar{G} \leq \frac{H(BL_0)}{H_{min}(s_1,\ldots,s_R)} .$$

Since the numerator, $H(BL_0)$, of the above bound is fixed, an efficient decision tree can be obtained by making the denominator, $H_{min}(s_1,\ldots,s_R)$, as large as possible during the construction. Furthermore, since

$$H_{min}(s_1,\ldots,s_R) = \min_i \; F(s_{i-1}, s_i),$$

in order to maximize $H_{min}(s_1,\ldots,s_R)$ it suffices to maximize $F(s_{i-1}, s_i)$ at each step of the construction. The above discussion motivates the following definition.

Definition 1: Suppose $0=s_0 < s_1 < \ldots < s_{R-1} < s_R$ be a given set of integers. An algorithm which maximizes $F(s_{i-1}, s_i)$, $i=1,\ldots,R$, during its construction is defined to be a generalized optimum testing algorithm of order $(s_1,\ldots,s_R)$ and is denoted by GOTA $(s_1,\ldots,s_R)$.

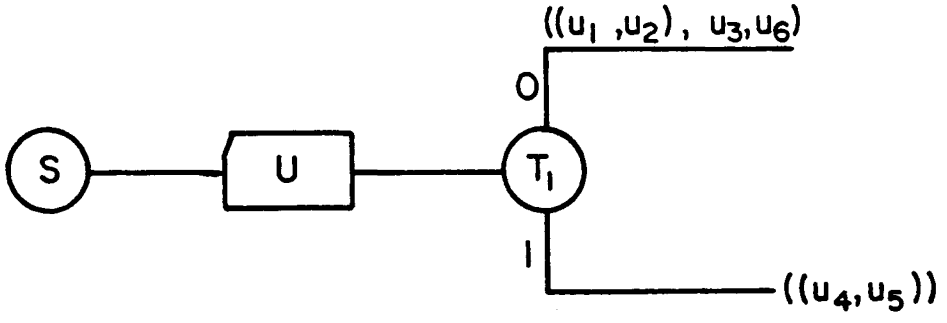
To construct GOTA $(s_1,\ldots,s_R)$, we first select a set of tests which maximize $F(0,s_1)$. Based on the choice of the above set of tests, the second set of tests is selected which maximizes $F(s_1,s_2)$. This procedure is continued until all the actions are identified. It should be noted that no tests are repeated in any path because repetition of a test in any path will produce a less

efficient algorithm. They may, however, appear in other paths. The above procedure does not necessarily provide us with an optimum algorithm since the selection of tests which maximize $F(s_{i-1},s_i)$ is conditioned upon the selection of the previous tests. The construction of the algorithm is illustrated by means of the following examples.

Example 4: In this example, we construct GOTA $(1,2,\ldots,s_R)$ for the problem posed in Example 1 for the many-to-one mapping ϕ_2. The efficiency measure is assumed to be the average execution time which is proportional to the average path length, $\overline{W}$, of the decision tree. We select the first test which maximizes $F(0,1)$. For any selection of the first test, $g(0,1)$ has the same value, 1, and, therefore, minimization of $H(BL_1)$ corresponds to the maximization of $F(0,1)$. The values of $H(BL_1)$ for all the available tests are given in the following table:

Tests	$H(BL_1)$ bits
T_1	0.5195
T_2	1.0612
T_3	1.2013
T_4	0.7899
T_5	1.2508

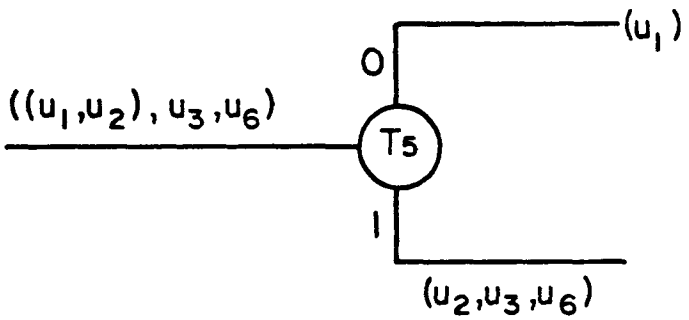
We select T_1 as the first test which corresponds to the smallest value of $H(BL_1)$ and yields the largest value of $F(0,1)$. The decision tree for GOTA $(1,2,\ldots,s_R)$ begins as



where u_1 and u_2 have been put into a paranthesis because they correspond to the same action. Now we select the second test so as to maximize F(1,2). Note that u_4 and u_5 correspond to the same action, therefore, the lower branch doesn't need to be pursued. Also, tests which do not distinguish at least two sets of objects are not used, and, therefore, g(1,2) has the same value regardless of the selection of the next test. Thus, maximization of F(1,2) corresponds to a minimization of $H(BL_2)$. The values of $H(BL_2)$ corresponding to the remaining tests is given in the following table:

Tests	$H(BL_2)$ bits
T_2	0.3181
T_3	0.3754
T_4	0.3754
T_5	0.3000

We, therefore, select T_5 and obtain the following tree

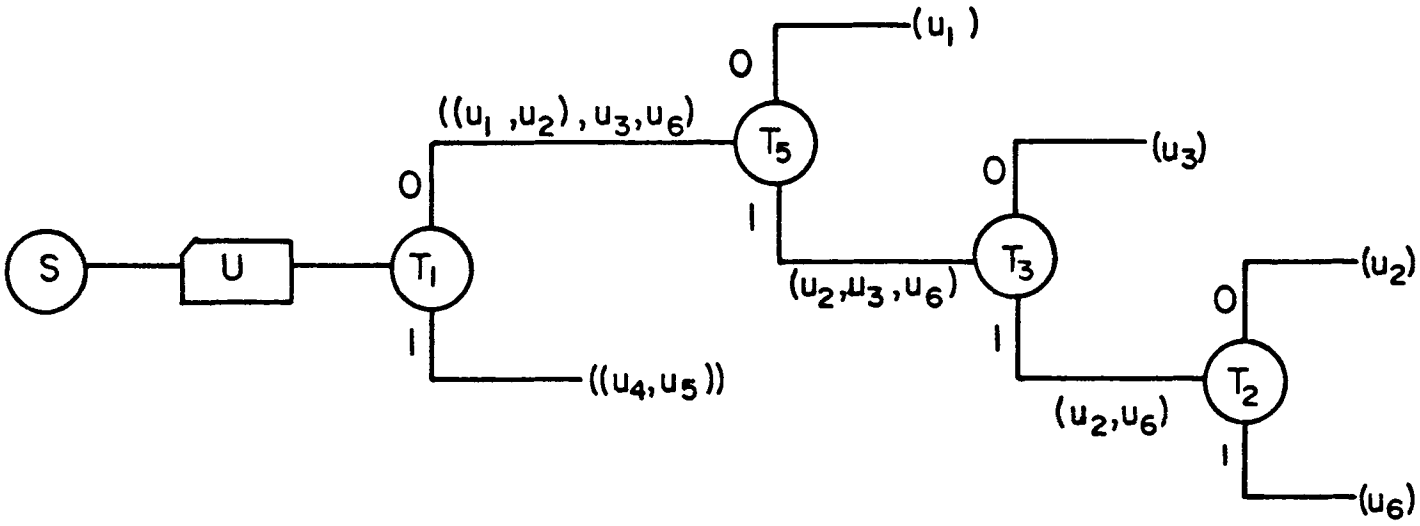


By a similar reasoning, g(2,3) is the same regardless of the selection of the next test and a minimization of $H(BL_3)$ maximizes F(2,3). The values of $H(BL_3)$ for the remaining tests are given by the following table:

Tests	$H(BL_3)$ bits
T_2	0.1377
T_3	0.1000
T_4	0.1000

Out of T_3 and T_4, we arbitrarily select the test T_3 as the next test.

Finally, we select T_2 to distinguish u_2 from u_6. The decision tree for GOTA $(1,2,\ldots,s_R)$ is given by



It can be easily shown that the above decision tree is the optimum solution with $\overline{W} = 1.7$. For this example the lower bound $\overline{W}_{Huff}$ is 1.55. This lower bound cannot be achieved here since proper tests are not available. Note that the algorithm outlined in [28] is also applicable to this problem. However, it is less efficient since its objective is to distinguish all the unknown objects and, in fact, it can be shown that its average path length is at least 2.3.

In the next example, we illustrate the construction of GOTA $(1,2,\ldots,s_R)$

and GOTA $(2,3,\ldots,s_Q)$ and compare their efficiencies.

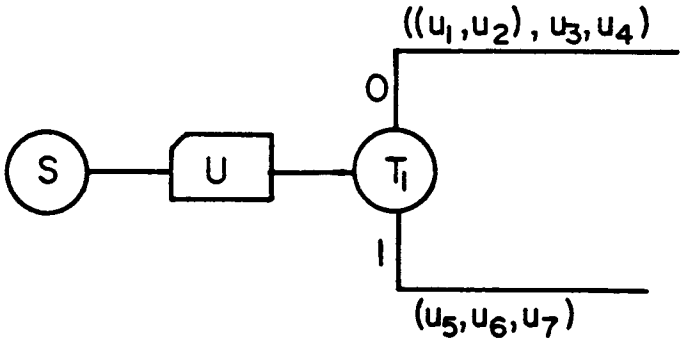
Example 5: Let us consider the following limited-entry decision table for U along with the probability measure $P_U(u)$ and the mapping ϕ.

T \ u	u_1	u_2	u_3	u_4	u_5	u_6	u_7
$P_U(u)$	0.10	0.60	0.01	0.02	0.10	0.05	0.12
$\phi(u)$	A_1	A_1	A_2	A_3	A_4	A_5	A_6
T_1	0	0	0	0	1	1	1
T_2	1	1	0	1	1	0	0
T_3	1	0	0	1	0	0	1
T_4	0	0	0	1	1	0	1
T_5	0	1	0	1	0	1	1

We again consider average execution time as our efficiency measure which is proportional to the average path length, $\overline{W}$. First we construct GOTA $(1,2,\ldots,s_R)$. As in Example 4, we first select the test which minimizes $H(BL_1)$, the values of which are provided in the following table for all the tests.

Tests	$H(BL_1)$ bits
T_1	0.6131
T_2	0.7735
T_3	1.0745
T_4	0.6594
T_5	1.1259

We select T_1 as the first test and the decision tree begins as



Next, we select the second set of tests, one for the upper branch and one for the lower branch, which minimizes $H(BL_2)$. We should point out that this selection can be done independently for each branch of the above tree, because after the selection of the first test, T_1, each branch becomes an independent problem. The minimization of $H(BL_2)$ can be attained by the individual minimizations of $H(BL_2|\text{Upper})$ and $H(BL_2|\text{Lower})$ which are the contributions of the upper and lower branches to $H(BL_2)$ respectively, i.e.,

$$\min_{T} H(BL_2) = \min_{T_i} H(BL_2|\text{Upper}) + \min_{T_j} H(BL_2|\text{Lower})$$

where

$$H(BL_2|\text{Upper}) = P_U(u_{123})\ E(u_{12}, u_3) + P_U(u_4)\ E(u_4)$$

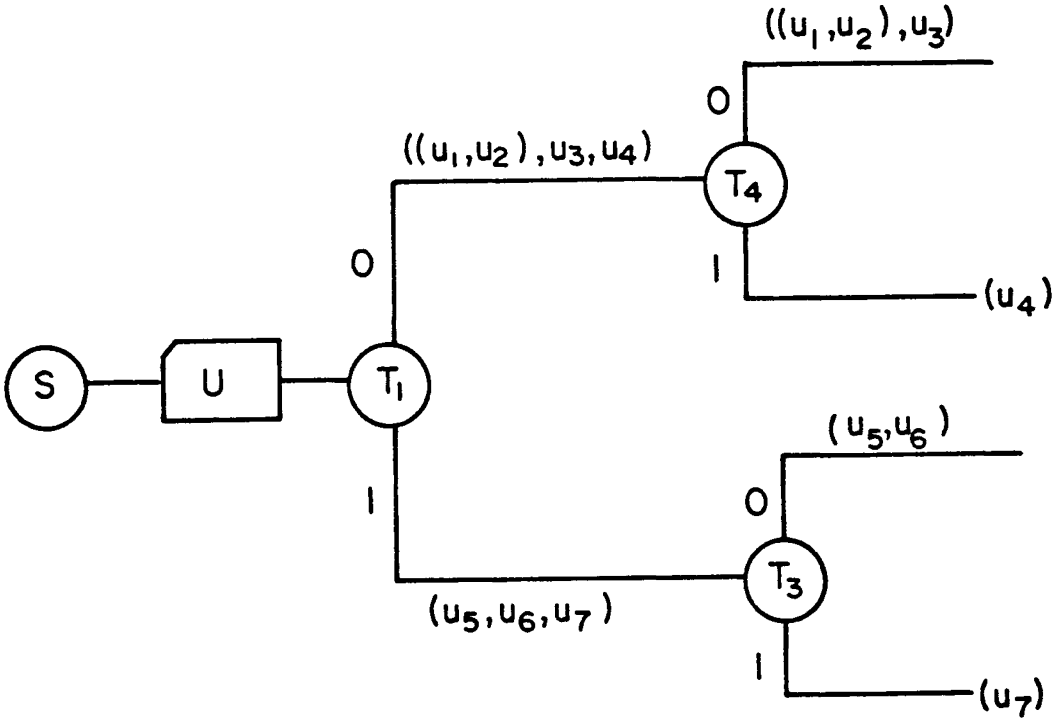
and

$$H(BL_2|\text{Lower}) = P_U(u_{56})\ E(u_5, u_6) + P_U(u_7)\ E(u_7)$$

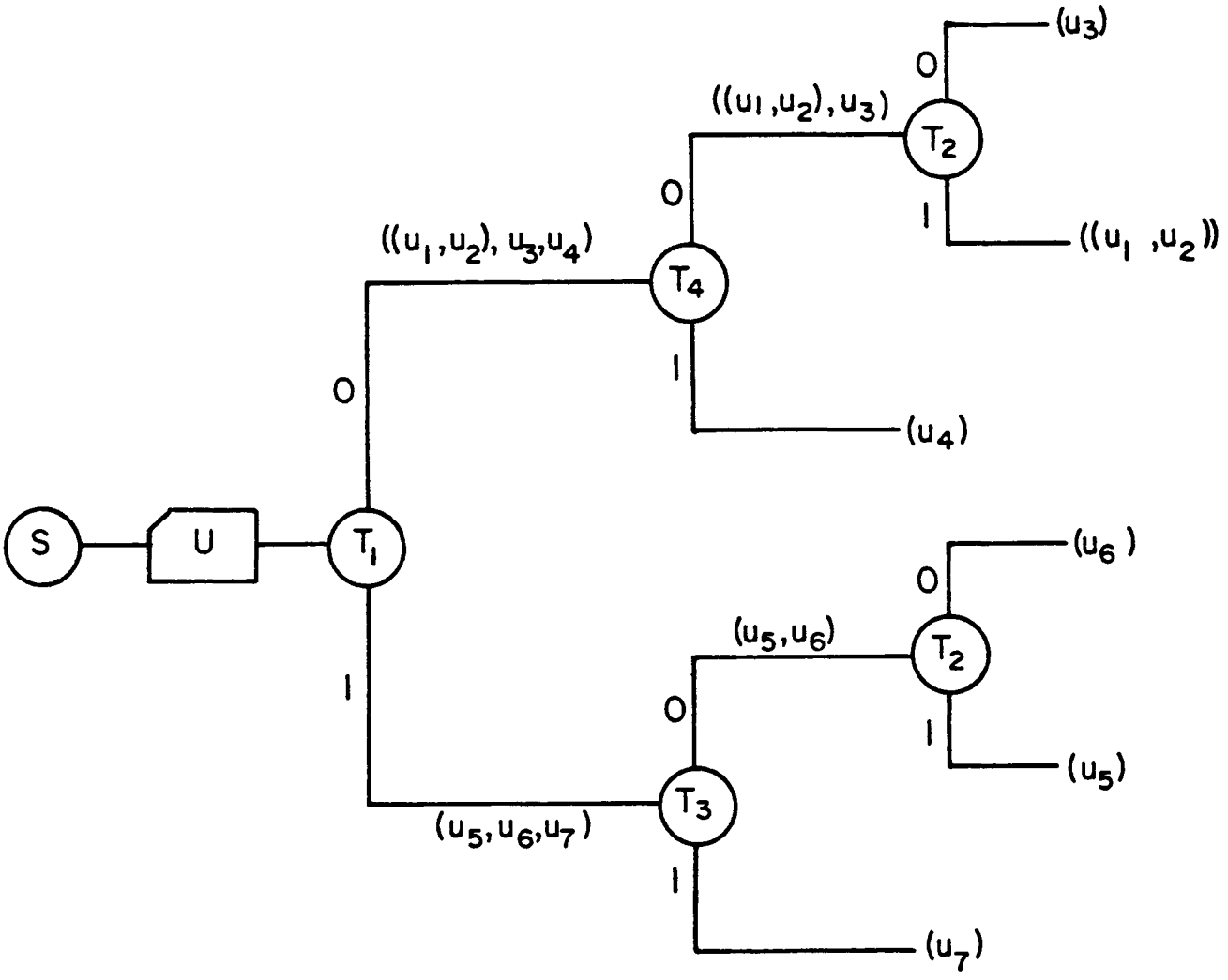
The values of $H(BL_2|\text{Upper})$ and $H(BL_2|\text{Lower})$ for the remaining tests are listed below.

Tests	$H(BL_2\|Upper)$ bits	$H(BL_2\|Lower)$ bits
T_2	0.1327	0.1485
T_3	0.1502	0.1377
T_4	0.0755	0.2187
T_5	0.1750	0.1485

We select T_4 as the test for the upper branch and T_3 as the test for the lower branch. The decision tree, therefore, becomes



The decision tree can now be easily completed and is given by



We note that GOTA $(1,2,\ldots,s_R)$ yields $\overline{W} = 2.86$.

Next, we construct GOTA $(2,3,\ldots,s_Q)$ which is expected to be more efficient due to Theorem 1. Since $s_1 = 2$, we must select tests which maximize $F(0,2)$ which is given by

$$F(0,2) = \frac{H(BL_0) - H(BL_2)}{g(0,2)}$$

where

$$g(0,2) = g(0,1) + g(1,2)$$

From the limited-entry decision table, we note that there is no single test which uniquely identifies any action. Therefore, g(0,2) is equal to two for all possible choices of tests at this step of the algorithm. Therefore, a maximization of F(0,2) corresponds to a minimization of $H(BL_2)$. For any selection of the first test, $H(BL_2)$ is minimized by minimizing $H(BL_2|\text{Upper})$ and $H(BL_2|\text{Lower})$ independently. These values are tabulated below for all possible choices of the first test.

A. First test T_1:

Tests	$H(BL_2\|\text{Upper})$ bits	$H(BL_2\|\text{Lower})$ bits
T_2	0.1327	0.1485
T_3	0.1502	0.1377
T_4	0.0755	0.2187
T_5	0.1750	0.1485

The minimum value of $H(BL_2)$ when the first test is T_1 is min $H(BL_2)$ = 0.0755 + 0.1377 = 0.2132 bits

B. First test T_2:

Tests	$H(BL_2\|\text{Upper})$ bits	$H(BL_2\|\text{Lower})$ bits
T_1	0.1485	0.1327
T_3	0.0391	0.4925
T_4	0.0391	0.0781
T_5	0.1485	0.1750

min $H(BL_2)$ = 0.1172 bits

C. First test T_3

Tests	$H(BL_2\|Upper)$ bits	$H(BL_2\|Lower)$ bits
T_1	0.2099	0.0781
T_2	0.4535	0.0781
T_4	0.3291	0.0829
T_5	0.2335	0.0829

min $H(BL_2)$ = 0.2880 bits

D. First test T_4

Tests	$H(BL_2\|Upper)$ bits	$H(BL_2\|Lower)$ bits
T_1	0.0755	0.2187
T_2	0.0391	0.0781
T_3	0.3291	0.0829
T_5	0.3029	0.2407

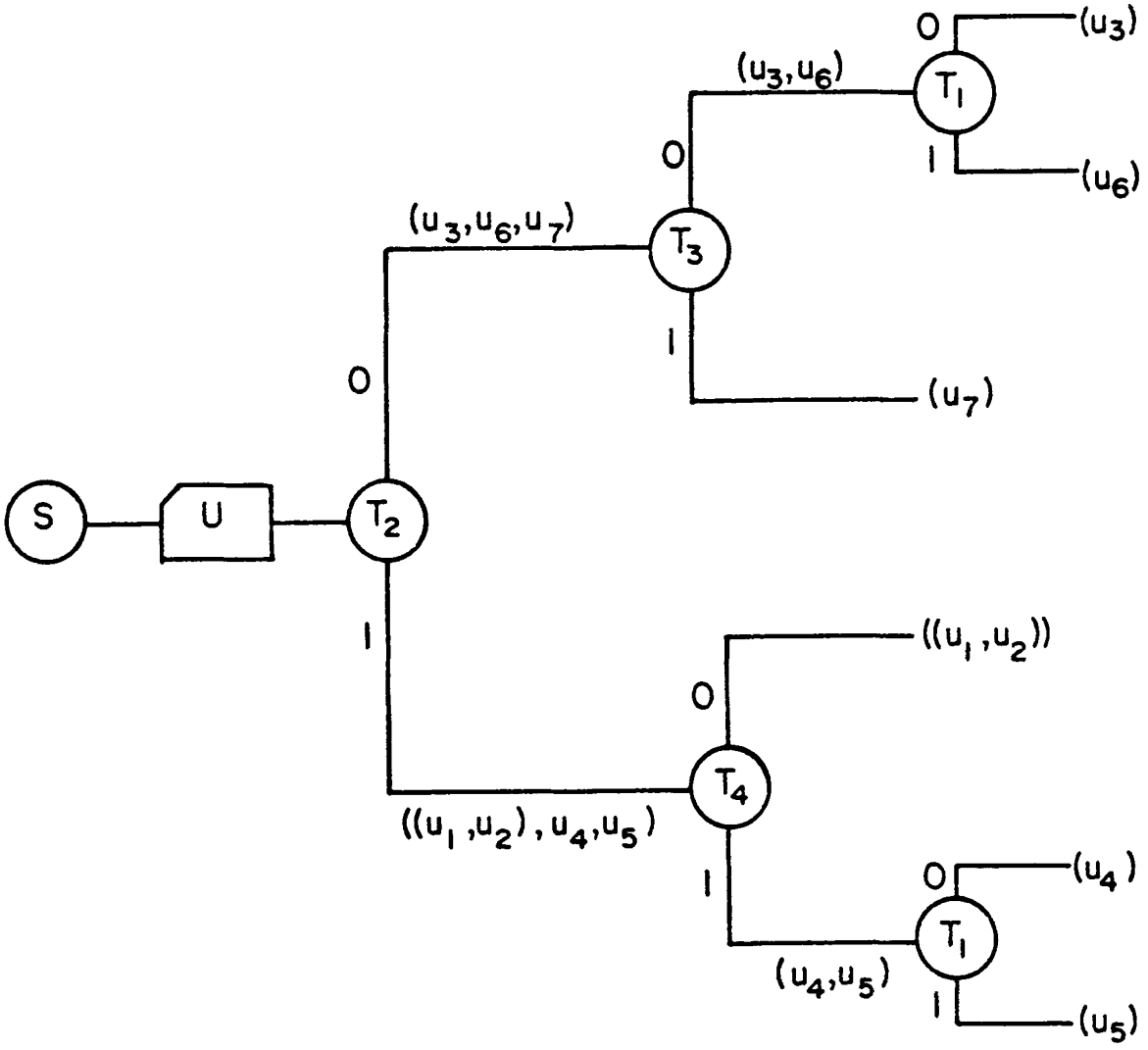
min $H(BL_2)$ = 0.1172 bits.

E. First test T_5

Tests	$H(BL_2\|Upper)$ bits	$H(BL_2\|Lower)$ bits
T_1	0.0484	0.2752
T_2	0.2000	0.2752
T_3	0.0484	0.2680
T_4	0.0484	0.2680

min $H(BL_2)$ = 0.3164 bits

Thus, the minimum value of $H(BL_2)$ is 0.1172 bits which corresponds to first test T_2 and subsequent tests T_3 (or T_4) and T_4 in the upper and lower branches respectively. We note that another choice which also gives the minimum value of $H(BL_2)$ is to have T_4 as the first test and T_2 as the subsequent test in both the branches. We may complete the above three trees by appropriately selecting the next set of tests, one of which is given below



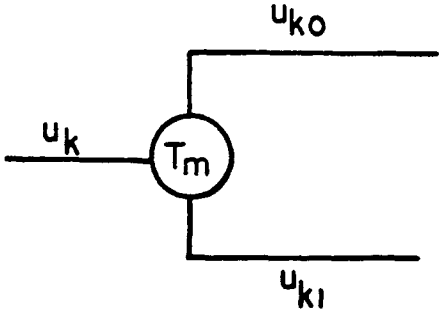
For the above decision tree, $\overline{W} = 2.18$ which is, as expected, more efficient than GOTA $(1,2,\ldots,s_R)$ and the improvement in $\overline{W}$ is 24%. It can easily be shown that this is an optimum solution for this problem. We should point out that the other two decision trees also provide the same efficiency.

In Examples 4 and 5, during the construction of GOTA $(1,2,\ldots,s_R)$, maximization of F(i-1, i) corresponded to the minimization of $H(BL_i)$ since g(i-1, i) was the same for all possible choices of tests at any step of the algorithm. In general, this result is true for the construction of GOTA $(1,2,\ldots,s_R)$ with the commonly used efficiency measures, average execution time and storage. Furthermore, at any stage of the construction of the tree further extension of each branch becomes an independent problem. Therefore, minimization of $H(BL_k)$ can be achieved by minimizing the $H(BL_k|\text{Upper})$ and $H(BL_k|\text{Lower})$ for each problem, as illustrated in Example 5.

We should point out that if ϕ is a one-to-one mapping, then it can be shown in a straightforward manner that the algorithms GOTA and OTA [28] provide the same decision trees. Therefore, we may conclude that GOTA $(1,2,\ldots,s_R)$ when ϕ is a one-to-one mapping provides the same decision tree as Massey's first-order optimal algorithm [18,19].

4.2 Tables With Dashes

Up to this point, we have concentrated on the construction of decision trees for decision tables when all the information is available. In other words, when T_m, m=1,...,M, is applied, the probability of the outcome d is always known. When all of these probabilities are not known, i.e., the decision table contains dashes, we must modify the above algorithm for the construction of decision trees. Let us consider a test T_m with binary outcomes which has a dash in the decision table corresponding to the object u_k. When T_m is applied to u_k, then with an unknown probability $\alpha_k P_U(u_k)$ the test outcome is zero and with probability $(1-\alpha_k)P_U(u_k)$ the test outcome is one as shown below

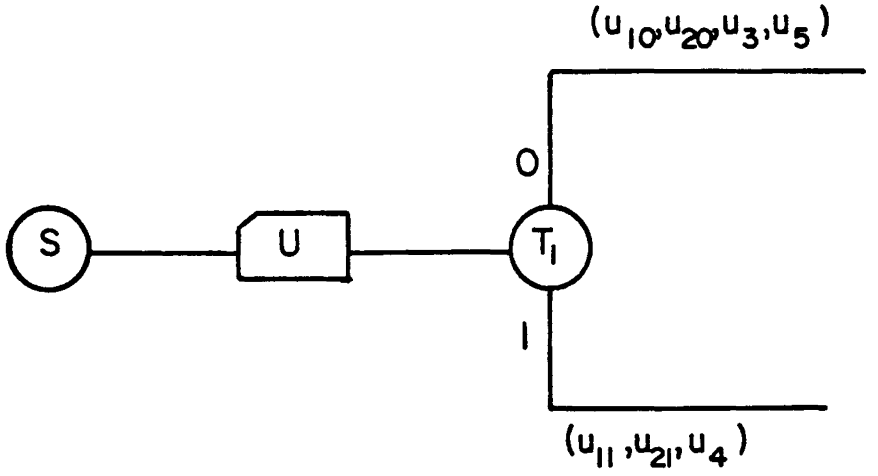


where $P_U(u_{k0}) = \alpha_k P_U(u_k)$ and $P_U(u_{k1}) = (1-\alpha_k)P_U(u_k)$. One possible approach is to arbitrarily set α_k equal to 0.5 and construct the GOTA as outlined above. Another approach is to obtain the values of the unknown α_k's for each T_m so as to minimize $F(s_{i-1}, s_i)$ at each step of the construction of GOTA. This approach, which is consistent with Pollack [8], discourages the use of the tests which have dashes in the decision table. In this approach, during the construction of the decision tree, a test with dashes in the decision table is used only if under the worst conditions its performance is better than all the remaining tests. We illustrate the second algorithm in the following example. Details regarding the computation of α_k's are provided in Appendix B.

Example 6: In this example, we consider the following limited-entry decision table alongwith the associated probability measure $P_U(u)$ and the mapping ϕ.

T \ u	u_1	u_2	u_3	u_4	u_5
$P_U(u)$	0.3	0.1	0.1	0.2	0.3
$\phi(u)$	A_1	A_2	A_3	A_3	A_4
T_1	-	-	0	1	0
T_2	0	1	1	0	1
T_3	1	1	0	-	1
T_4	1	0	1	0	1

The efficiency measure to be considered is average execution time and we construct GOTA $(1,2,\ldots,s_R)$. Since in this example, tests T_1 and T_3 contain dashes, we compute the values of α_k's which minimize $F(0,1)$ and this, in turn, corresponds to the maximization of $H(BL_1)$ for the tests T_1 and T_3. These values of α_k's are employed to calculate $H(BL_1)$ for tests T_1 and T_3. For the test T_1, we have



where

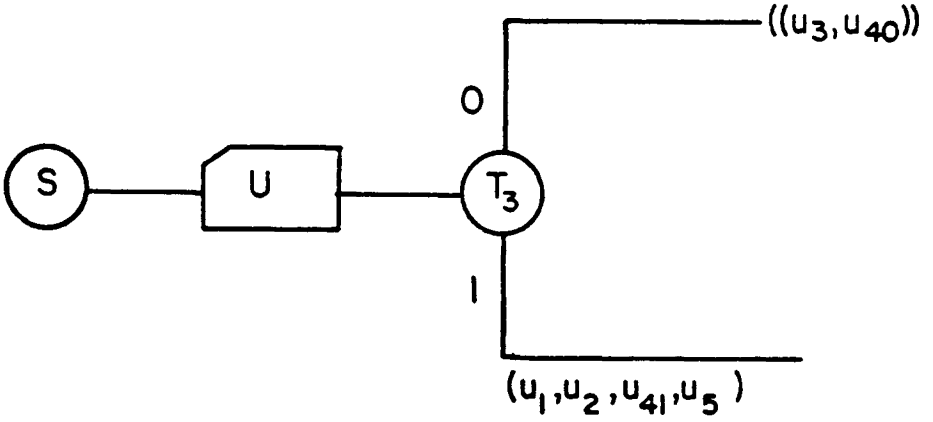
$$P_U(u_{10}) = \alpha_1 P_U(u_1),\ P_U(u_{11}) = (1-\alpha_1)\ P_U(u_1),$$

$$P_U(u_{20}) = \alpha_2 P_U(u_2) \text{ and } P_U(u_{21}) = (1-\alpha_2)\ P_U(u_2).$$

From Appendix B, it follows that the values of α_1 and α_2 which maximize $H(BL_1)$ for T_1 are given by

$$\alpha_1 = \alpha_2 = \frac{p_3+p_5}{p_3+p_4+p_5} = 0.667$$

and the corresponding maximum value of $H(BL_1)$ is 1.6201 bits. In a similar manner, for test T_3, we have

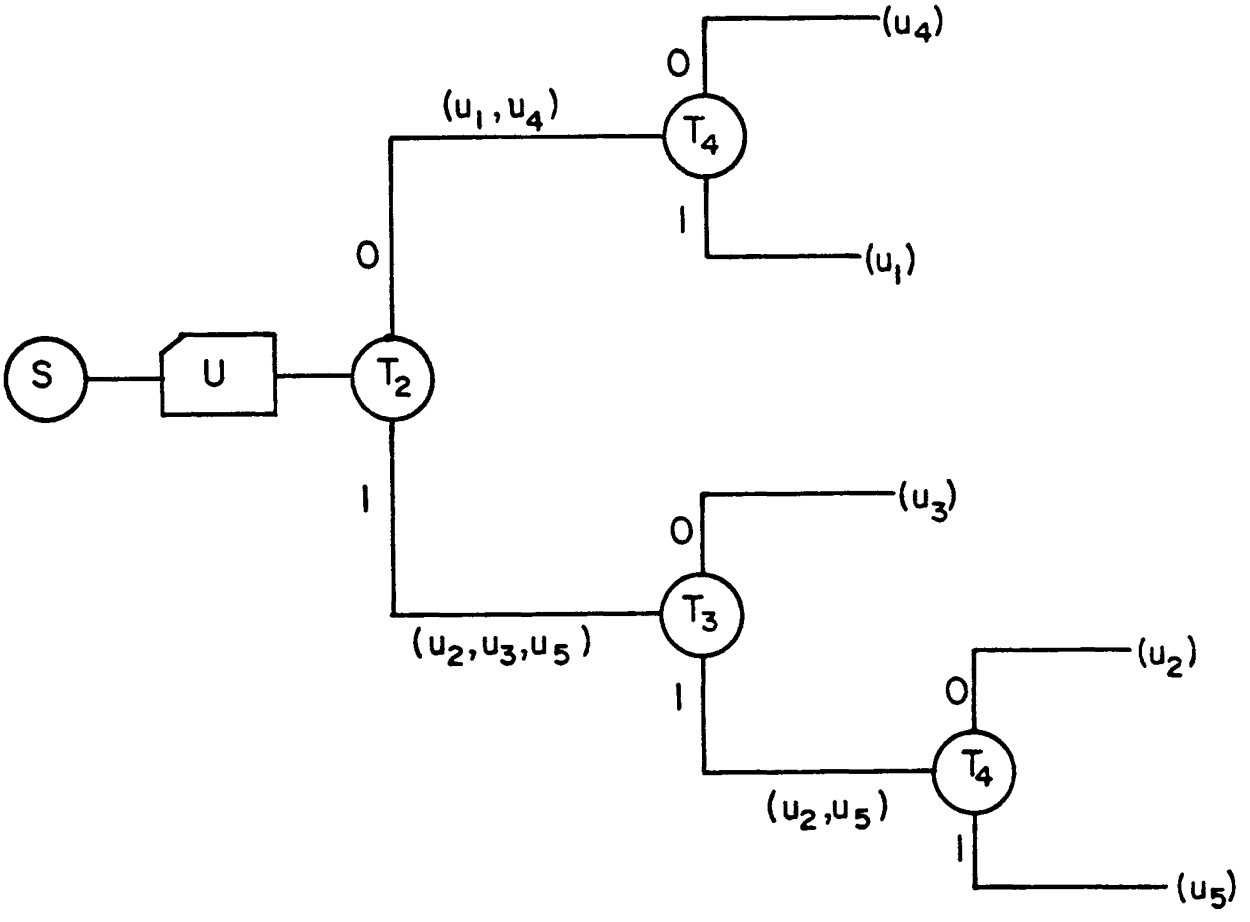


where $P_U(u_{40}) = \alpha_4 P_U(u_4)$ and $P_U(u_{41}) = (1-\alpha_4)P_U(u_4)$.

The value of α_4 which maximizes $H(BL_1)$ and the maximum $H(BL_1)$ are given by 0 and 1.7016 bits respectively. The values of $H(BL_1)$ for T_2 and T_4 are obtained in the usual manner. $H(BL_1)$ for all the tests are given below:

Tests	$H(BL_1)$ bits
T_1	1.6201
T_2	1.1710
T_3	1.7016
T_4	1.2886

Clearly, T_2 is the first test of the decision tree. Proceeding in a straightforward fashion, the complete decision tree, is



For the above decision tree, $\overline{W} = 2.4$ which is the optimum for this example. If we had selected α_k's to be 0.5, we would get the same decision tree in this case.

In the next section, we discuss the complexity of the construction of GOTA $(s_1,\ldots,s_R)$.

5. COMPLEXITY OF THE CONSTRUCTION OF GOTA

In the previous section, we proposed a systematic algorithm for the construction of efficient decision trees. In this section, we study the complexity of this construction. The complexity measure considered here, $GC(s_1,\ldots,s_R)$, is the number of computations of the function $F(s_{i-1}, s_i)$ during the construction of GOTA $(s_1,\ldots,s_R)$. Since $GC(s_1,\ldots,s_R)$ depends upon the specific problem under consideration, it is not possible to compute its exact value. Therefore, we obtain an upper bound on $GC(s_1,\ldots,s_R)$ by evaluating the complexity of the worst case which occurs when a tree that is complete at all levels is constructed. Since a test used at any decision node may not be used at subsequent decision nodes of the same subtree, a simple counting argument provides us the following result:

$$GC(s_1,\ldots,s_R) \leq \sum_{j=1}^{R} \prod_{i=s_{j-1}+1}^{s_j} (M-i+1)^{D^{i-1}} \tag{12}$$

where recall that M is the number of tests.

Another considerably smaller, upper bound on the complexity measure is obtained in the special case when the function g satisfies

$$g(s_{i-1}, s_i) = g(s_{i-1}, s_{i-1}+1) + g(s_{i-1}+1, s_{i-1}+2) +\ldots+ g(s_i-1, s_i)$$

and that $g(s_i-1, s_i)$ does not depend upon the selection of the tests at BL_{s_i-1}. Then, once the tests prior to BL_{s_i-1} have been specified, $g(s_{i-1}, s_i)$ remains constant for each possible selection of the tests at the current level BL_{s_i-1}. Thus, maximization of $F(s_{i-1}, s_i)$ corresponds to the minimization of $H(BL_{s_i})$. This minimization can be performed independently for each subtree corresponding to each selection of tests prior to BL_{s_i-1}, as in Example 5. Therefore, an upper bound is given by

$$GC(s_1,\ldots,s_R) \leq \sum_{j=1}^{R} D^{s_{j-1}} (M-s_j+1) \prod_{i=s_{j-1}+1}^{s_j-1} (M-i+1)^{D^{i-1}} \tag{13}$$

where $\prod_{i=s_{j-1}+1}^{s_j-1} (M-i+1)^{D^{i-1}} = 1$

whenever $s_j - s_{j-1} = 1$.

Example 7: Let us compute the upper bounds for the constructions in Example 5. We have, M = 5 and D = 2. Using (13), the upper bound on GC(1,2,3) is 25. If we use (12), the upper bound is 102. The actual complexity is 19. For GC(2,3), the upper bounds are 52 and 161 respectively whereas the actual complexity is 46.

6. SUMMARY AND CONCLUSIONS

In this paper, we have presented a systematic approach to the construction of efficient decision trees from decision tables which may include "don't care" entries based on information theoretic concepts. The basic philosophy in our approach is the same one as used in [28] in which the upper bound on the efficiency measure is minimized at each step of the construction of decision trees. Such heuristic procedures are important in practice since the construction of optimum decision trees is an NP-complete problem [31-32] in many cases.

We observe that the systematic procedure presented in this paper provides us with a trade-off between the complexity of the construction of the decision tree and the upper bound on the efficiency measure. In other words, a smaller upper bound on $\overline{G}$ may be achieved by choosing larger values of $(s_i - s_{i-1})$'s and thereby increasing the complexity of the construction of GOTA $(s_1,\ldots,s_R)$. In most cases, this provides us a more efficient decision tree. However, in some rare instances this may not occur as found in Example 9 of [28]. This does not contradict Theorem 1 since it provides the relationship between upper bounds on $\overline{G}$ for the same decision tree. However, the construction of GOTA for different sets of s_i's may lead to different decision trees. The importance of Theorem 1 is to provide a clue for the selection of the set of s_i's.

Now we suggest the general procedure for the construction of efficient decision trees whenever a lower bound on $\overline{G}$, $LB\overline{G}$, can be computed.

(1) Compute $LB\overline{G}$.

(2) Construct GOTA $(1,2,\ldots,s_R)$ and calculate the associated efficiency measure $\overline{G}\ (1,2,\ldots,s_R)$. If $\overline{G}\ (1,2,\ldots,s_R)$ is close to $LB\overline{G}$, accept GOTA $(1,2,\ldots,s_R)$ as the solution. Otherwise continue.

(3) Construct GOTA $(2,3,\ldots,s_Q)$ and calculate the associated efficiency measure $\overline{G}\ (2,3\ldots,s_Q)$. If $\overline{G}\ (2,3\ldots,s_Q)$ is close to $LB\overline{G}$, accept GOTA $(2,3\ldots,s_Q)$ as the solution. If $\overline{G}\ (2,3.,,,.s_Q)$ is close to $\overline{G}\ (1,2,\ldots,s_R)$, we may conclude that we are near the optimum value of $\overline{G}$ and accept the algorithm with smaller efficiency measure as the solution. Otherwise continue with the selsection of other values of s_i's until an acceptable solution is achieved.

ACKNOWLEDGEMENT

We would like to thank Dr. S. J. Hong of I.B.M. for his comments and helpful discussion on the subject.

REFERENCES

[1] Shannon, C.E., A mathematical Theory of communication. Bell Syst. Tech. J. 27 (July 1948), 379-423.

[2] Shannon, C.E., A mathematical theory of communication, Bell Syst. Tech. J. 27 (Oct. 1948), 623-656.

[3] Garey, M.R., Optimal binary identification procedures. SIAM J. Appl. Math 23, 2 (Sept. 1972), 173-186.

[4] Schumacher, H., and Sevcik, K.C., The Synthetic approach to decision table conversion, Comm. ACM 19, 6 (June 1976), 343-351.

[5] Reinwald, L.T., and Soland, R,M., Conversion of limited entry decision tables to optimal computer programs I: Minimum average processing time. J. ACM 13, 3 (July 1966), 339-358, II: Mimimum storage requirement. J. ACM 14, 4 (Oct. 1967), 742-755.

[6] Montalbano, M., Tables, flowcharts and program logic. IBM Systems J. (Sept. 1962), 51-63.

[7] Egler, J.F., A procedure for converting logic table conditions into an efficient sequence of test instructions. Comm. ACM 6, 9 (Sept. 1963), 510-514.

[8] Pollack, S.L., Conversion of limited entry decision tables to computer programs. Comm. ACM 8, 11 (Nov. 1965), 677-682.

[9] King, P.J.H., Decision tables. Computer J. 10, 2 (Aug. 1967), 135-142.

[10] Shwayder, K., Conversion of limited entry decision tables to computer programs--A proposed modification to Pollack's algorithm. Comm. ACM 14, 2 (Feb. 1971), 69-73.

[11] Ganapathy, S., Information theory applied to decision tables. M. Tech. Th., Indian Institute of Technology, Kanpur, India, (July 1969).

[12] King, P.J.H., Conversion of decision tables to computer programs by rule mask techniques. Comm. ACM 9, 11 (Nov. 1966), 769-801.

[13] Muthukrishnan, C.R., and Rajaraman, V., On the conversion of decision tables to computer programs. Comm. ACM 13, 6 (June 1970), 347-351.

[14] Ibramshah, M., and Rajaraman, V., A fast rule mask algorithm for the conversion of decision tables to computer programs. Techn. Rept. Computer Centre, Indian Institute of Technology, Kanpur, India, 1971.

[15] Kirk, H.W., Use of decision tables in Computer programming. Comm. ACM 8, 1 (Jan. 1965), 41-43.

[16] Shwayder, K., Extending the information theory approach to converting limited-entry decision tables to computer programs. Comm. ACM 17, 9 (Sept. 1974), 532-537.

[17] Goel, D.K., Random test generation for fault detection and diagnosis. Ph.D. dissertation, School of Computer and Information Science, Syracuse University, Syracuse, N.Y., 1978.

[18] Massey, J.L., An information-theoretic approach to data-processing algorithms, Presented at 1974 IEEE Int. Symp. Info. Th., Notre Dame, Indiana, Oct. 1974.

[19] Massey, J.L., Topics in Discrete Information Processing. Unpublished manuscript, Dept. of Electrical Engineering, University of Notre Dame, Indiana, 1976.

[20] Verhelst, M., The conversion of limited-entry decision tables to optimal and near optimal flowcharts: two new algorithms. Comm. ACM 15, 11 (Nov. 1972), 974-980.

[21] Ganapathy, S. and Rajaraman, V., Information theory applied to the conversion of decision tables to computer programs. Comm. ACM 16, 9 (Sept. 1973), 532-539.

[22] Alster, T.M., Heuristic algorithms for constructing near-optimal decision trees. Report No. UIUCDCE-R-71-474, Department of Computer Science, University of Illinois, Urbana, IL., Aug. 1971.

[23] Bayes, A.T. A dynamic programming algorithm to optimise decision table code, Australian Computer T. 4 (May 1973), 77-79.

[24] Fisher, D.L., Data documentation and decision tables. Comm. ACM, 18 (Jan. 1965), 26-31.

[25] Jarvis, J.M. An analysis of programming via decision table compilers. SIGPLAN Notices (ACM Newsletter) 6,8 (Sept. 1971), 30-32.

[26] Pooch, U.W. Translation of decision tables. Computing Surveys 6, (June 1974), 125-51.

[27] Rabin, T., Conversion of limited-entry decision tables into optimal decision trees: fundamental concepts. SIGPLAN Notices (ACM Newsletter) 6, (Sept. 1971), 68-71.

[28] Faria, J.M., Hartmann C.R.P., Gerberich C.L., and Varshney P.K., An information theoretic approach to the construction of efficient decision trees, Tech. Report 1-80, School of Comput. and Info. Sc., Syracuse University, Syracuse, N.Y. 13210, Jan. 1980. Also presented at the 1981 IEEE Int. Symp. Info. Th., Santa Monica, California, Feb. 1981.

[29] Michalski, R.S., Designing extended entry decision tables and optimal decision trees using decision diagrams, Report No. UIUCDCS-R-78-898, Department of Computer Science, University of Illinois, Urbana, Ill., March 1978.

[30] Yasui, T., Conversion of Decision Tables into Decision Trees, Report No. UIUCDCS-R-72-501, Department of Computer Science, University of Illinois, Urbana, ILL., Feb. 1972.

[31] Comer, D. and Sethi, R., The complexity of trie index construction. J. ACM 24, 3 (July 1977), 428-440.

[32] Hyafil, L. and Rivest, R.L., Constructing optimal binary decision trees is NP-complete. Information Processing Letters 5, 1 (May 1976), 15-17.

[33] Gallager, R.G., Information Theory and Reliable Communication. New York: Wiley, 1968.

[34] Rao, C.R., Linear Statistical Inference and Its Applications, New York: Wiley, 1974.

APPENDICES

Appendix A: In this appendix, we show that $F(s_{i-1}, s_i) \geq 0$. Since $g(s_{i-1}, s_i)$ is always positive, it is sufficient to show that $H(BL_{s_{i-1}}) - H(BL_{s_i}) \geq 0$.

However,

$$H(BL_{s_{i-1}}) - H(BL_{s_i}) = \sum_{j=s_{i-1}}^{s_i-1} [H(BL_j) - H(BL_{j+1})]$$

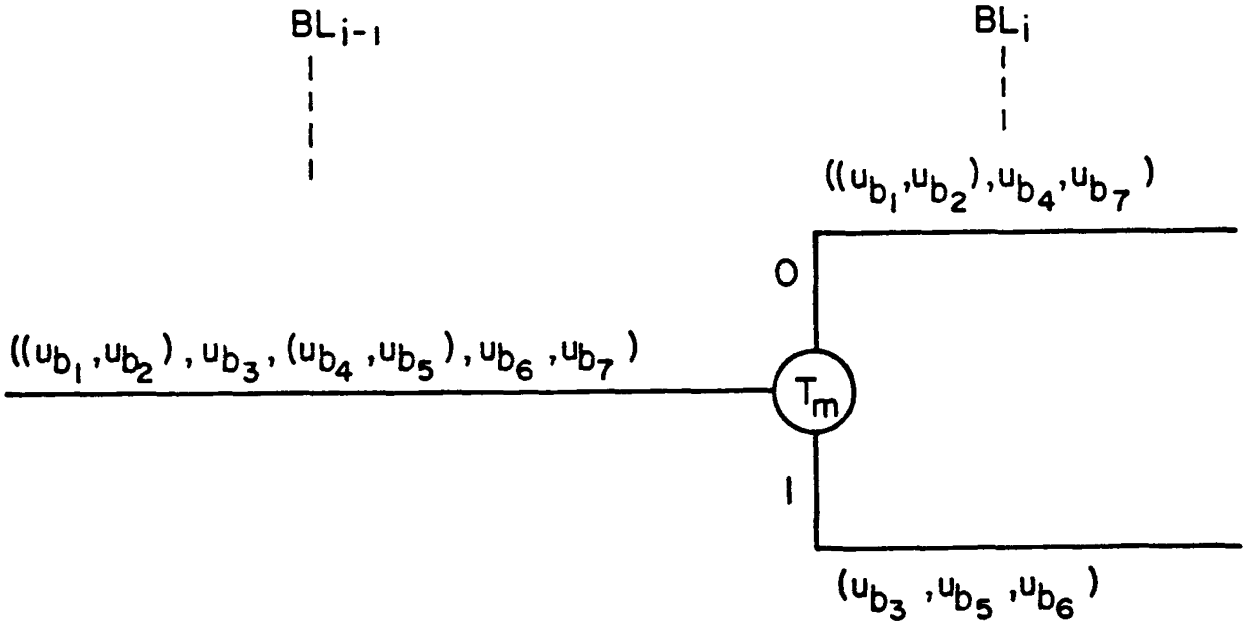
Therefore, we only need to show that

$$H(BL_{i-1}) - H(BL_i) \geq 0 \quad \text{for all i's.}$$

Theorem A: $H(BL_{i-1}) - H(BL_i) \geq 0$ for $i=1,2,\ldots,s_R$.

Proof: For the sake of clarity, we present the proof for the binary case only which can be generalized in a straightforward manner.

In general, there are several branches and associated subtrees at BL_{i-1}. Consider a typical nonterminal branch at BL_{i-1} as shown below



where u_{b_1} and u_{b_2} (also u_{b_4} and u_{b_5}) have been put into a paranthesis because $\phi(u_{b_1}) = \phi(u_{b_2})$ (and $\phi(u_{b_4}) = \phi(u_{b_5})$). The corresponding contribution $(H(BL_{i-1}) - H(BL_i))_{T_m}$ of this branch to $H(BL_{i-1}) - H(BL_i)$ is given by

$$(H(BL_{i-1}) - H(BL_i))_{T_m} = P_U(u_{b_1b_2\ldots b_7})\ E(u_{b_1b_2}, u_{b_3}, u_{b_4b_5}, u_{b_6}, u_{b_7})$$

$$- P_U(u_{b_1b_2b_4b_7})\ E(u_{b_1b_2}, u_{b_4}, u_{b_7}) - P_U(u_{b_3b_5b_6})\ E(u_{b_3}, u_{b_5}, u_{b_6})$$

Notice that $H(BL_{i-1}) - H(BL_i)$ is a summation of terms similar to $(H(BL_{i-1}) - H(BL_i))_{T_m}$ and, therefore, it suffices to show that $(H(BL_{i-1}) - H(BL_i))_{T_m} \geq 0$.

For notational convenience, we denote $P_U(u_{b_i})$ by p_i and also,

$$P = p_1+p_2+p_3+p_4+p_5+p_6+p_7$$

$$P_0 = p_1+p_2+p_4+p_7$$

$$P_1 = p_3+p_5+p_6$$

Now, we may express

$$-(H(BL_{i-1}) - H(BL_i))_{T_m} = -(p_1+p_2)\log\frac{p_1+p_2}{P_0} - p_4\log\frac{p_4}{P_0} - p_7\log\frac{p_7}{P_0}$$

$$- p_3\log\frac{p_3}{P_1} - p_5\log\frac{p_5}{P_1} - p_6\log\frac{p_6}{P_1}$$

$$+ (p_1+p_2)\log\frac{p_1+p_2}{P} + p_3\log\frac{p_3}{P} + (p_4+p_5)\log\frac{p_4+p_5}{P}$$

$$+ p_6\log\frac{p_6}{P} + p_7\log\frac{p_7}{P}$$

$$= (p_1+p_2)\log\left\{\frac{(p_1+p_2)P_0/P}{(p_1+p_2)}\right\} + p_3\log\left\{\frac{p_3P_1/P}{p_3}\right\}$$

$$+ p_4\log\left\{\frac{(p_4+p_5)P_0/P}{p_4}\right\} + p_5\log\left\{\frac{(p_4+p_5)P_1/P}{p_5}\right\}$$

$$+ p_6\log\left\{\frac{p_6P_1/P}{p_6}\right\} + p_7\log\left\{\frac{p_7P_0/P}{p_7}\right\}$$

$$= (p_1+p_2)\log\frac{\beta_1}{(p_1+p_2)} + \sum_{i=3}^{7} p_i\log\frac{\beta_{i-1}}{p_i} \qquad \text{(A1)}$$

where

$$\sum_{i=1}^{6}\beta_i = \frac{(p_1+p_2)P_0}{P} + \frac{p_3P_1}{P} + \frac{(p_4+p_5)P_0}{P} + \frac{(p_4+p_5)P_1}{P} + \frac{p_6P_1}{P} + \frac{p_7P_0}{P}$$

$$= \frac{(p_1+p_2+p_4+p_5+p_7)P_0}{P} + \frac{(p_3+p_4+p_5+p_6)P_1}{P}$$

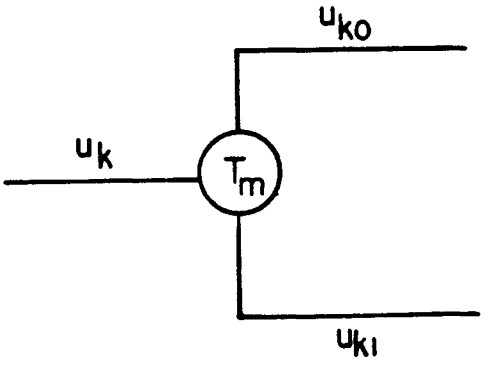
$$\leq P_0 + P_1 = P = \sum_{j=1}^{7} p_j$$

Thus, using (1e.6.1) of [34] we conclude

$$(H(BL_{i-1})-H(BL_i))_{T_m} \geq 0. \qquad \text{(A2)}$$

Furthermore, equality is achieved in (A2) if all the arguments of the logarithm are 1 in (A1).

Appendix B: In this appendix, mathematical details pertaining to the don't care case, discussed in Secion 4, are presented. Recall that when there are dashes in the limited-entry decision table, the maximum of $F(s_{i-1}, s_i)$ cannot be obtained because the probabilities of some of the test outcomes are unknown. Throughout this appendix, we only consider the case when the tests have binary outcomes. The results can, however, be generalized in a straightforward manner. We also restrict our attention to the construction of GOTA $(1,2,\ldots,s_R)$. Let us consider a test T_m which has a dash in the decision table corresponding to the object u_k. Then as before,

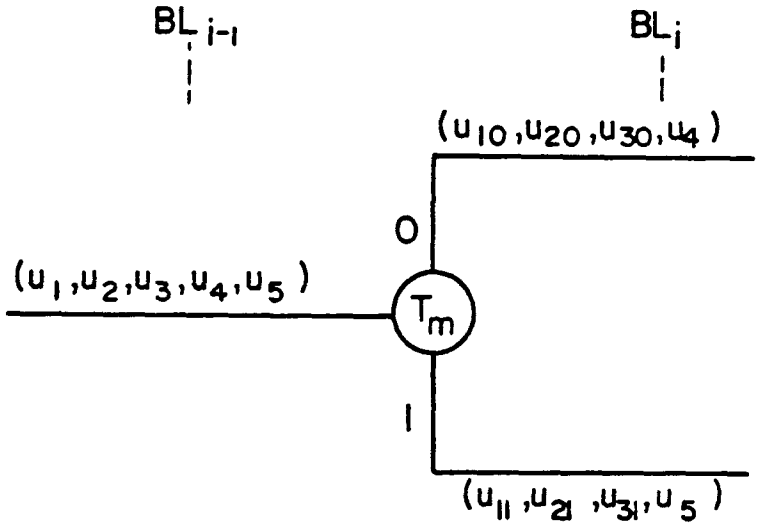


where $P_U(u_{k0}) = \alpha_k P_U(u_k)$ and $P_U(u_{k1}) = (1-\alpha_k)\, P_U(u_k)$ and α_k $(0 \leq \alpha_k \leq 1)$ is an unknown.

Assume that during the construction of GOTA $(1,2,\ldots,s_R)$, the next step is the selection of tests between BL_{i-1} and BL_i. This selection is made so

as to maximize F(i-1, i). During this selection, some of the tests may have dashes in the decision table. To calculate F(i-1, i) for this situation, we first find values of α_k's which minimize F(i-1, i). If g(i-1, i) is independent of α_k's, it suffices to maximize $H(BL_i)$. This can be done by maximizing the contribution of individual subtrees to $H(BL_i)$. For clarity in presentation, we consider the following special cases.

Case I: Consider the following subtree.



In terms of unknown coefficients α_i, i=1,2,3, the contribution to $H(BL_i)$ may be expressed as

$$(H(BL_i))_{T_m} = -\sum_{j=1}^{3} \{ \alpha_j p_j \log \frac{\alpha_j p_j}{P_0} + (1-\alpha_j)p_j \log \frac{(1-\alpha_j)p_j}{P_1} \}$$

$$+ p_4 \log \frac{p_4}{P_0} + p_5 \log \frac{p_5}{P_1} .$$

where $p_i = P_U(u_i)$, i=1,2,...,5 and $P_0 = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 + p_4$ and $P_1 = (1-\alpha_1)\, p_1 + (1-\alpha_2)\, p_2 + (1-\alpha_3)p_3 + p_5$. The α_i's are obtained so as to maximize $(H(BL_i))_{T_m}$, i.e. by solving the equations

$$\frac{\partial}{\partial \alpha_j} \left[(H(BL_i))_{T_m} \right] = 0, \quad j=1,2,3. \tag{B1}$$

This gives

$$\alpha_j = \frac{p_4}{p_4+p_5} \quad , \quad j=1,2,3.$$

In particular, if $p_4 = p_5 = 0$, then the solution of equations (B1) is $\alpha_1 = \alpha_2 = \alpha_3$. Next, we want to show that the function $(H(BL_i))_{T_m}$ is concave ($\cap$) in the entire region, $R=\{0 \le \alpha_j \le 1, j=1,2,3\}$. In order to prove this, it is sufficient to show that the following matrix of the second-order partial derivatives is negative definite in R.

$$\left[\frac{\partial^2}{\partial \alpha_j \partial \alpha_\ell} (H(BL_i))_{T_m} \right] = - \begin{bmatrix} \frac{1}{\alpha_1(1-\alpha_1)p_1} - \frac{1}{P_0} - \frac{1}{P_1} & - \frac{1}{P_0} - \frac{1}{P_1} & - \frac{1}{P_0} - \frac{1}{P_1} \\ - \frac{1}{P_0} - \frac{1}{P_1} & \frac{1}{\alpha_2(1-\alpha_2)p_2} - \frac{1}{P_0} - \frac{1}{P_1} & - \frac{1}{P_0} - \frac{1}{P_1} \\ - \frac{1}{P_0} - \frac{1}{P_1} & - \frac{1}{P_0} - \frac{1}{P_1} & \frac{1}{\alpha_3(1-\alpha_3)p_3} - \frac{1}{P_0} - \frac{1}{P} \end{bmatrix}$$

Therefore, we need to show that

(i) The determinant of Z is negative.

(ii) All diagonal elements of Z are negative.

(iii) All principal minors have negative determinants.

To prove (i), we observe that

$$\det(Z) = - \left[1 - \left(\frac{1}{P_0} + \frac{1}{P_1} \right) \sum_{j=1}^{3} \alpha_j(1-\alpha_j)p_j \right]$$

and thus it is sufficient to show that

$$\frac{1}{P_0} + \frac{1}{P_1} < \frac{1}{\sum_{j=1}^{3} \alpha_j(1-\alpha_j)p_j} \tag{B2}$$

Since

$$P_0 > \sum_{j=1}^{3} \alpha_j p_j \quad \text{and} \quad P_1 > \sum_{j=1}^{3} (1-\alpha_j)p_j \quad , \tag{B3}$$

we have,

$$\frac{1}{P_0} + \frac{1}{P_1} < \frac{1}{\sum_{j=1}^{3} \alpha_j p_j} + \frac{1}{\sum_{j=1}^{3} (1-\alpha_j) p_j} = \frac{\sum_{j=1}^{3} p_j}{(\sum_{j=1}^{3} \alpha_j p_j)(\sum_{j=1}^{3} (1-\alpha_j) p_j)} \qquad \text{(B4)}$$

Thus it remains to show that the RHS of (B4) is less than the RHS of (B2). The difference,

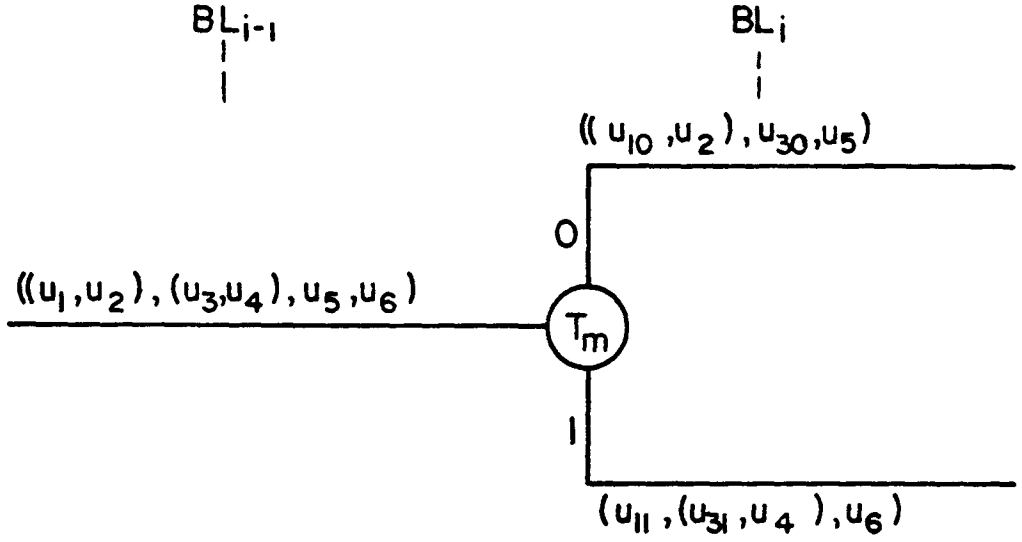
$$\frac{\sum_{j=1}^{3} p_j}{(\sum_{j=1}^{3} \alpha_j p_j)(\sum_{j=1}^{3} (1-\alpha_j) p_j)} - \frac{1}{\sum_{j=1}^{3} \alpha_j (1-\alpha_j) p_j}$$

$$= - \frac{\sum_{j=1}^{3} \sum_{\ell=1}^{3} p_j p_\ell (\alpha_j - \alpha_\ell)^2}{(\sum_{j=1}^{3} \alpha_j p_j)(\sum_{j=1}^{3} (1-\alpha_j) p_j)(\sum_{j=1}^{3} \alpha_j (1-\alpha_j) p_j)}$$

is obviously negative.

From (B3), it is easy to see that (ii) holds. The proof of (iii) is analogous to the proof of (i). This discussion can be generalized to any arbitrary number of unknown objects.

Case II: Consider the following subtree



$(H(BL_i))_{T_m}$ can be expressed as

$$(H(BL_i))_{T_m} = -\left[(\alpha_1 p_1+p_2)\log \frac{\alpha_1 p_1+p_2}{P_0} + \alpha_3 p_3 \log \frac{\alpha_3 p_3}{P_0} + p_5 \log \frac{p_5}{P_0}\right.$$

$$+ (1-\alpha_1)p_1 \log \frac{(1-\alpha_1)p_1}{P_1} + ((1-\alpha_3)p_3+p_4) \log \frac{((1-\alpha_3)p_3+p_4)}{P_1}$$

$$\left. + p_6 \log \frac{p_6}{P_1}\right]$$

where $P_0 = \alpha_1 p_1+p_2+\alpha_3 p_3+p_5$ and $P_1 = (1-\alpha_1)p_1 + (1-\alpha_3)p_3 + p_4+p_6$. We now set

$$\alpha_1 p_1+p_2 = \gamma_1 q_1 \quad \text{and} \quad (1-\alpha_1)p_1 = (1-\gamma_1)q_1 ,$$

$$\alpha_3 p_3 = \gamma_3 q_3 \quad \text{and} \quad (1-\alpha_3)p_3+p_4 = (1-\gamma_3)q_3.$$

After this transformation, $(H(BL_i))_{T_m}$ has the same form as in Case I. Therefore, the maximum is obtained at

$$\gamma_1 = \gamma_3 = \frac{p_5}{p_5+p_6}$$

But

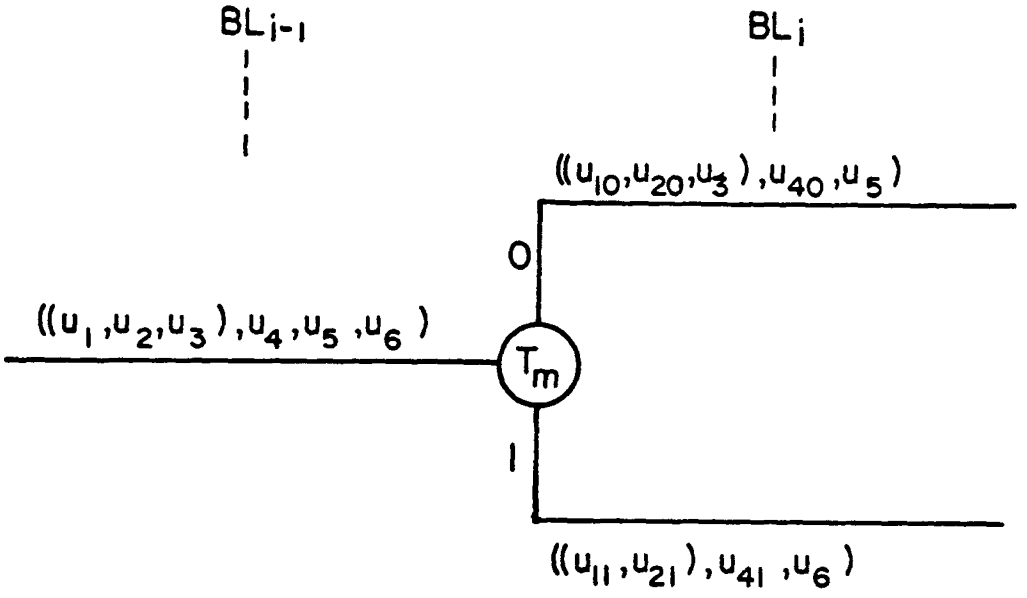
$$\alpha_1 = \gamma_1 - (1-\gamma_1)\frac{p_2}{p_1}$$

and

$$\alpha_3 = \gamma_3 \frac{p_3+p_4}{p_3}$$

We should notice that α_1 can be negative whereas α_3 can be greater than one. Since $(H(BL_i))_{T_m}$ is concave with respect to γ_1 and γ_3 and the transformation to α's is linear, the maximum is obtained at the boundary, i.e. at $\alpha_1 = 0$ whenever $\gamma_1 - ((1-\gamma_1)p_2)/p_1$ is negative and $\alpha_3 = 1$ whenever $(\gamma_3(p_3+p_4))/p_3$ is greater than one. In the particular case when $p_5=p_6=0$, if we can find a $\gamma = \gamma_1 = \gamma_3$ such that α_1 and α_3 both lie between 0 and 1, then any such value of α_1 and α_3 will provide the maximum of $(H(BL_i))_{T_m}$. If no such value of γ can be obtained, then, as before, the maximum is obtained by setting $\alpha_1 = 0$ and $\alpha_3 = 1$.

Case III: In this last case we consider the following subtree whose $(H(BL_i))_{T_m}$ can be expressed as



$$(H(BL_i))_{T_m} = -\ [(\alpha_1p_1+\alpha_2p_2+p_3)\ \log\ \frac{\alpha_1p_1+\alpha_2p_2+p_3}{P_0} + \alpha_4p_4\ \log\ \frac{\alpha_4p_4}{P_0} +$$

$$p_5\ \log\ \frac{p_5}{P_0}\ + \{(1-\alpha_1)p_1+(1-\alpha_2)p_2\}\ \log\ \frac{\{(1-\alpha_1)p_1+(1-\alpha_2)p_2\}}{P_1}$$

$$+\ (1-\alpha_4)p_4\ \log\ \frac{(1-\alpha_4)p_4}{P_1}\ +\ p_6\ \log\ \frac{p_6}{P_1}\ ]\quad ,$$

where $P_0 = \alpha_1p_1 + \alpha_2p_2 + p_3 + \alpha_4p_4 + p_5$ and $P_1 = (1-\alpha_1)p_1 + (1-\alpha_2)p_2 + (1-\alpha_4)p_4 + p_6$. We now set

$$\alpha_1p_1 + \alpha_2p_2 + p_3 = \gamma_1q_1 \quad \text{and} \quad (1-\alpha_1)p_1 + (1-\alpha_2)p_2 = (1-\gamma_1)q_1$$

and obtain $(H(BL_i))_{T_m}$ in the same form as in Case I. Therefore, the maximum is obtained at

$$\gamma_1 = \alpha_4 = \frac{p_5}{p_5+p_6}\ .$$

From the above transformation $q_1 = p_1+p_2+p_3$ and

$$\alpha_1 \frac{p_1}{q_1} + \alpha_2 \frac{p_2}{q_1} + p_3 = \gamma_1 \quad . \tag{B5}$$

If a solution of the above equation, satisfying $0 \leq \alpha_i \leq 1$, $i=1,2$, exists, then any such solution will give the maximum value of $(H(BL_i))_{T_m}$. If $\gamma_1 - p_3 < 0$ then no solution of (B5) satisfies $0 \leq \alpha_i \leq 1$, $i=1,2$, and in this case the maximum is obtained at $\alpha_i = 0$, $i=1,2$. If $p_1/q_1 + p_2/q_1 < \gamma_1 - p_3$ then again no solution of (B5) satisfies $0 \leq \alpha_i \leq 1$, $i=1,2$, and in this case the maximum is obtained at $\alpha_i = 1$, $i=1,2$. Whenever $p_5 = p_6 = 0$, the solution is analogous to the Case II.